SHP-VWNS
13404124

L-force *Controls*

Softwarehandbuch

Software Manual

VisiWinNET Smart

Erste Schritte

Getting started

Lenze

Dokumenthistorie

Materialnummer	Version			Beschreibung
13201079	1.0	03/2007	TD29	Erstausgabe
13227673	2.0	12/2007	TD29	Beschreibung eines Beispielprojekts ergänzt
13236341	3.0	04/2008	TD29	Beschreibung zum Kommunikationstreiber PLC Designer ergänzt
13292736	4.0	06/2009	TD29	An Software-Version 6.3 angepasst und um Kapitel ”Beispielprojekt” erweitert
13342001	5.0	06/2010	TD29	Abschnitt ”Skripte”, ”Einschränkungen bei Geräten der Reihe EL 100” und ”Übersetzungsmanager” ergänzt
13351708	6.0	11/2010	TD29	An Software-Version 6.4 angepasst
13404124	6.1	03/2012	TD29	An Software-Version 6.5 angepasst

Tipp!

Informationen und Hilfsmittel rund um die Lenze-Produkte finden Sie im Download-Bereich unter

http://www.Lenze.com

1 Vorwort und Allgemeines 6
1.1 Lieferumfang 6
1.2 Lizenzbedingungen 6
1.3 Systemanforderungen 7
1.4 Einschränkungen bei Geräten der Reihe EL 100 7

2 Installation 8

3 Bedienung 9
3.1 Grundlagen 9
3.1.1 Programm starten 9
3.1.2 Beschreibung der Programmoberfläche 9
3.1.3 Hilfen für den Anwender 11
3.1.4 Projekt öffnen 12
3.1.5 Projekt anlegen 13
3.1.6 Projekt speichern 14
3.1.7 Projekt konvertieren 15
3.1.8 Programm beenden 15
3.2 Entwicklungsumgebung 16
3.2.1 Projektexplorer 16
3.2.2 Formulardesigner, Toolbox und Eigenschaftenfenster 18
3.2.3 Ausgabefenster 20
3.2.4 Tabelleneditor 21
3.2.5 Eigenschaftenseite 22
3.2.6 Menüleiste 23
3.2.7 Werkzeugleiste ”Standard” 25
3.2.8 Werkzeugleiste ”Format” 25
3.3 Emulator für Windows® CE-Projekte 26
3.3.1 Emulator starten 27
3.3.2 Emulator-Fenster 28
3.3.3 Variablen-Simulationsfenster 30

4 Beispielprojekt 33
4.1 Beispielprojekt anlegen 33
4.2 Kommunikationskanal erstellen 38
4.3 Variablen browsen 40
4.4 Eine Visualisierung erstellen 41
4.4.1 Formular erstellen und Werte darstellen 41
4.4.2 Weitere Formulare erstellen und zwischen diesen navigieren 44

4.5 Benutzer verwalten ... 49
4.5.1 Benutzerverwaltungssystem festlegen ... 51
4.5.2 Rechte anlegen ... 52
4.5.3 Benutzergruppen anlegen ... 53
4.5.4 Benutzer anlegen ... 55
4.5.5 Dialoge für die Benutzerverwaltung einsetzen ... 57
4.6 Mehrsprachige Projekte erstellen (Sprachumschaltung) ... 60
4.6.1 Sprachumschaltung aufrufen ... 61
4.6.2 Benutzertexte anlegen ... 62
4.6.3 Benutzertexte einem Steuerelement vom Typ ”Label” zuweisen ... 63
4.6.4 Sprache während der Laufzeit umschalten ... 65
4.6.5 Weitere Sprachen definieren ... 66
4.6.6 Dynamische Texte verwenden (Variablen anzeigen) ... 68
4.7 Alarmssystem konfigurieren ... 71
4.7.1 Prozessvariablen definieren ... 72
4.7.2 Alarmgruppen anlegen ... 73
4.7.3 Alarme anlegen ... 74
4.7.4 Alarm-Quittierungsart festlegen ... 76
4.7.5 Alarme visualsieren ... 77
4.7.6 Alarmsystem testen ... 78
4.7.7 Alarm-Anzeigen filtern ... 79
4.8 Prozessdaten aufzeichnen (Archivsystem) ... 80
4.9 Visualisierungsrezepte definieren ... 84
4.10 Informationen drucken ... 88

5 Konfiguration ... 91

5.1 Projekteinstellungen ... 91
5.1.1 Globale Projekteinstellungen ... 91
5.1.2 Allgemeine Projekteigenschaften ... 91
5.1.3 Auswahl der Systeme ... 91
5.1.4 Erweiterte Projekteigenschaften ... 92
5.1.5 Laufzeitverhalten ... 93
5.1.6 Spracheinstellungen ... 93
5.2 Kommunikation ... 94
5.2.1 Projekt über LAN auf das EL1xx übertragen ... 94
5.2.2 CAN-Verbindung konfigurieren über Codestellen ... 96
5.2.3 CAN-Verbindung konfigurieren über PDOs ... 99
5.2.4 MPI-Verbindung konfigurieren ... 103
5.2.5 L-force Logic (CoDeSys)-Verbindung konfigurieren ... 108

6 Anhang 112
6.1 sdf-Datei aus einer Symboltabelle erzeugen 112
6.2 awl-Datei aus einem Datenbaustein erzeugen 113
6.3 sym-Datei im PLC-Designer erzeugen 116
6.4 Skripte 118
6.4.1 Grundlagen 118
6.4.2 Fehlersuche in Funktionen 123
6.4.3 Beispiel-Funktionen 124
6.4.4 Einschränkungen in der Programmierschnittstelle von VisiWinNET Smart 126
6.5 Übersetzungsmanager 127
6.5.1 Texte exportieren 128
6.5.2 Exportdatei schützen 130
6.5.3 Texte importieren 132

7 Stichwortverzeichnis 134

1 Vorwort und Allgemeines

VisiWinNET® Smart ist die leistungsfähige Entwicklungsumgebung für Bedieneinheiten der Serie EL100 (HMIs) oder Industrie-PCs (IPCs) von Lenze.

Mit VisiWinNET® Smart steht Ihnen ein Tool zur Verfügung, mit dem Sie auf komfortable Weise Visualisierungsapplikationen erstellen können.

Die Applikationen können Sie schließlich in VisiWinNET® Smart in das Bedieneinheit-spezifische Datenformat umwandeln (kompilieren) und in die entsprechende Bedieneinheit übertragen.

1.1 Lieferumfang

- DVD “VisiWinNET® Smart”
- Hardware-Dongle für Lizenzierung
- Lizenzbedingungen

Hinweis!

Überprüfen Sie nach Erhalt der Lieferung sofort, ob der Lieferumfang mit den Warenbegleitpapieren übereinstimmt. Für nachträglich reklamierte Mängel übernehmen wir keine Gewährleistung.

Reklamieren Sie

- erkennbare Transportschäden sofort beim Anlieferer.
- erkennbare Mängel / Unvollständigkeit sofort bei der zuständigen Lenze-Vertretung.

1.2 Lizenzbedingungen

Sowohl die Entwicklungsumgebung als auch das Laufzeitsystem unterliegen Lizenzbedingungen und sind gegen Missbrauch geschützt.

- Entwicklungsumgebung: Hardware-Dongle
- Laufzeitsystem: an Hardware gekoppelt

Hinweis!

VisiWinNET® Smart ist ohne Hardware-Dongle nur im ”Demo-Modus” mit folgender Einschränkung nutzbar:

- Nur ein Projekt mit dem vorgegebenen Namen ”VWNTest” kann angelegt und bearbeitet werden.

Im Demo-Modus steht in der Titelzeile das Wort ”Demo”.

1.3 Systemanforderungen

Um mit VisiWinNET® Smart arbeiten zu können, sind nach unserer Erfahrung folgende Anforderungen an Hard- und Software zu erfüllen:

Hardware	minimal	empfohlen
Computer	IBM-kompatibler PC	
CPU-Familie	Intel Pentium/Celeron	
CPU-Taktfrequenz	500 MHz	1 GHz oder mehr
Arbeitsspeicher	256 MB	512 MB oder mehr
Grafik	800 x 600 Pixel, High Color (16 Bit)	1280 x 1024 Pixel, True Color (32 Bit)
Freier Festplattenplatz	200 MB für Entwicklungssystem 280 MB für .Net-Framework (falls noch nicht installiert)	
DVD-Laufwerk	Nur für die Installation	
Schnittstellen	• Ethernet- oder CAN-Schnittstelle zur Kommunikation mit dem Zielgerät (alternativ freier COM- oder USB-Anschluss). • Freier USB-Anschluss für Hardware-Dongle	

Software	
Betriebssystem	Windows Server, XP SP2 oder Vista
Applikationen	Microsoft .NET Framework (im Lieferumfang, falls nicht vorhanden)
Optionale Applikationen *)	• Microsoft .NET Compact Framework 2.0 SP1, • Microsoft .NET Framework SDK 2.0 • PDF-Reader (für Anwendungshandbücher) • Internet-Browser (für Online-Hilfe) • Microsoft ActiveSync.

*) Optionale Software-Produkte, die auf dem Installationsrechner nicht zur Verfügung stehen, können kostenlos über die Internetseiten der entsprechenden Hersteller bezogen werden. Falls diese Produkte nicht installiert werden, stehen einige erweiterte Funktionen von VisiWinNET® Smart nicht zur Verfügung. Für die Installation wird weiterer freier Speicherplatz auf der Festplatte benötigt.

1.4 Einschränkungen bei Geräten der Reihe EL 100

Bei HMIs der Reihe EL 100 gibt es aus Performance-Gründen folgende Einschränkungen in VisiWinNET® Smart:

Funktion	EL 106 ... EL 110s	EL 105 / EL 103 ECO
Maximale Anzahl Variablen	1000	500
Minimale Abtastzeit einer Kommunikationsgruppe	100 ms	200 ms
Maximale Anzahl Alarme	2000	1000
Alarmhistorie, binärer Ringpuffer	mit maximal 1000 Alarmen	mit maximal 500 Alarmen
Trends	• max. 20 Trends • min. Abtastzeit 500 ms • max. 50 Trenddateien zur Laufzeit • min. Cachegröße 10 • 2000 Samples pro Datei	• max. 10 Trends • min. Abtastzeit 1000 ms • max. 50 Trenddateien zur Laufzeit • min. Cachegröße 10 • 2000 Samples pro Datei
Protokollierung	System ist komplett deaktiviert und kann auch vom Anwender nicht wieder aktiviert werden	
Rezepte	• max. 500 Variablen pro Rezept • max. 50 Rezeptdateien zur Laufzeit	• max. 250 Variablen pro Rezept • max. 50 Rezeptdateien zur Laufzeit

2 Installation

Hinweis!

Für die Installation der Software sind Administratorrechte erforderlich!

So gehen Sie vor:

1. Legen Sie die VisiWinNET® Smart-DVD in Ihr Laufwerk.

   Wenn die Autostart-Funktion Ihres Laufwerks aktiviert ist, wird das Installationsprogramm automatisch gestartet.

   Falls sie deaktiviert ist, starten Sie das Installationsprogramm über einen Doppelklick auf die Datei ”setup.exe” im Programmverzeichnis der Installations-DVD.

2. Folgen Sie den Anweisungen der Installationsdialoge. Sie wechseln zum nächsten Dialog durch einen Klick auf die Schaltfläche **Weiter**.

   Sobald VisiWinNET® Smart erfolgreich installiert ist, wird Ihnen das gemeldet. Sollten während der Installation Fehler auftreten, wird diese abgebrochen. Konsultieren Sie in dem Fall den Lenze-Service.

3. Klicken Sie auf die Schaltfläche **Fertigstellen**.

3 Bedienung

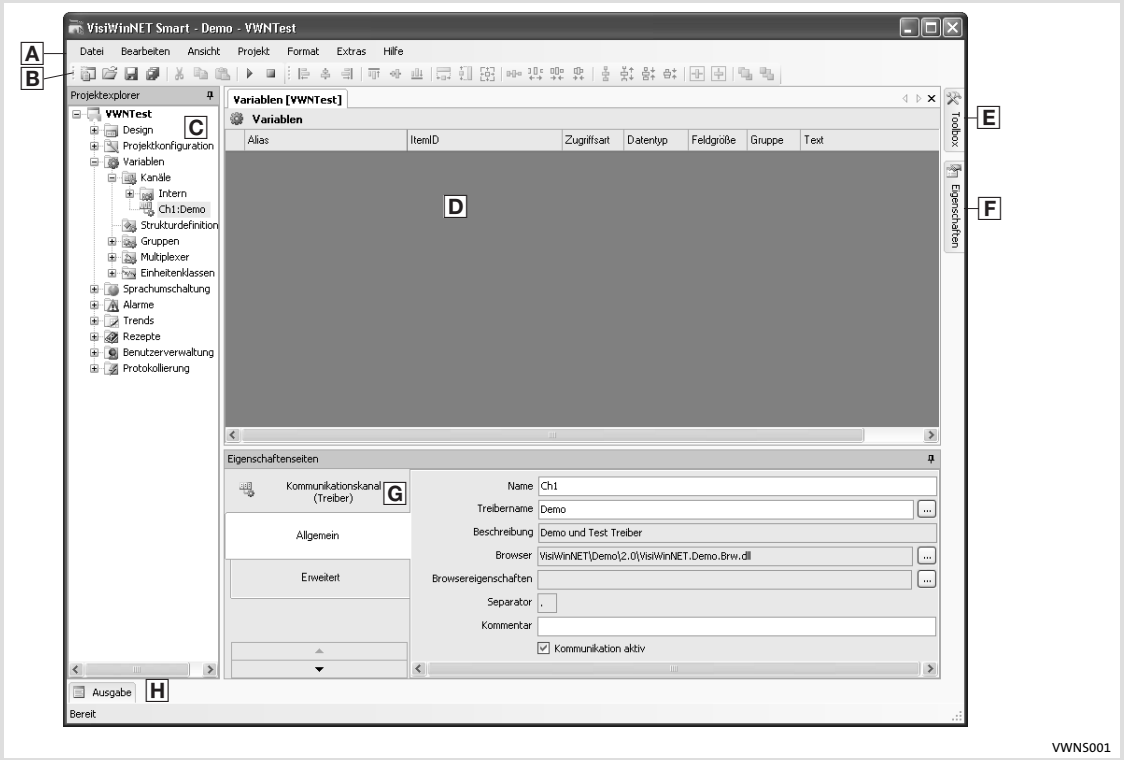
3.1 Grundlagen

3.1.1 Programm starten

So gehen Sie vor:

1. Stecken Sie den Hardware-Dongle auf.
2. Klicken Sie im **Startmenü** auf **Alle Programme** → **Lenze** → **VisiWinNET 2005** → **VisiWinNET Smart**.
   Der VisiWinNET® Smart-Bildschirm wird angezeigt.

3.1.2 Beschreibung der Programmoberfläche



- A Menüleiste
- B Symbolleiste
- C Projektexplorer
- D Clientbereich
- E Toolbox-Fenster
- F Eigenschaften-Fenster (Tools)
- G Eigenschaftenseiten (Objektlisten)
- H Ausgabefenster

Beschreibung der Programmoberflächen-Elemente

- Menüleiste A

  Jedes Menü beinhaltet eine Reihe thematisch zusammen gehörender Menübefehle. Durch Anwählen eines Menübefehls wird eine Funktion gestartet oder ein Untermenü mit weiteren Menübefehlen geöffnet.

- Symbolleiste B

  Über die Schaltflächen der Symbolleiste können einige Funktionen schnell (ohne Umweg über die Menüs) aufgerufen werden. Der Umfang der Symbolleiste kann vom Anwender nicht angepasst werden.

- Projektexplorer C

  Der Aufbau und die Funktion des Projektbaums sind weitgehend mit dem WINDOWS-Explorer vergleichbar. An oberster Stelle (erste Ebene) steht der Projektordner. Er beinhaltet (feste) Unterordner für die strukturierte Ablage der Objekte.

- Clientbereich D

  Im Clientbereich werden, in Abhängigkeit des im Projektexplorer gewählten Objekts, die zugehörige Objektliste oder das gewählte Formular (Laufzeitdialog) angezeigt.

- Toolbox E

  Die Toolbox beinhaltet die Steuerelemente für Formulare (Designs) des Laufzeitsystems.

- Eigenschaftenfenster (Formular-Steuerelemente) F

  Hier legen Sie die Eigenschaften zu dem Steuerelement fest, das im Clientbereich, in einem Formular, markiert ist.

- Eigenschaftenseiten G

  In den Eigenschaftenseiten legen Sie die Eigenschaften zu dem Objekt fest, das im Clientbereich, in einer Objektliste, markiert ist.

- Ausgabefenster H

  Systemmeldungen anzeigen.

Hinweis!

In diversen Bereichen können Sie mit der rechten Maustaste auf ein Objekt klicken und damit ein Kontextmenü öffnen, das häufig verwendete Befehle anbietet.

3.1.3 Hilfen für den Anwender

Die Anwenderdokumentation für VisiWinNET® Smart umfasst folgende Komponenten:

- Online-Hilfe

  VisiWinNET® Smart beinhaltet eine ausführliche Online-Hilfe. Sie erhalten Zugriff auf die Online-Hilfe über:

  – Menü **Hilfe** → **Hilfe** (Gesamthilfe)
  – Schaltfläche **Hilfe** in einem Dialog (kontextbezogene Hilfe)
  – Taste **[F1]** zum Beispiel auf einem Steuerelement in einem Formular, auf einer Steuerelement-Eigenschaft (kontextbezogene Hilfe) oder auf einer Eigenschaftenseite für Projektdefinitionen (Kanäle, Texte, Alarme usw.)

  Voraussetzung für den Aufruf der Online-Hilfe ist, dass ein Internet-Browser installiert ist.

- Anwender-Handbücher (PDFs)

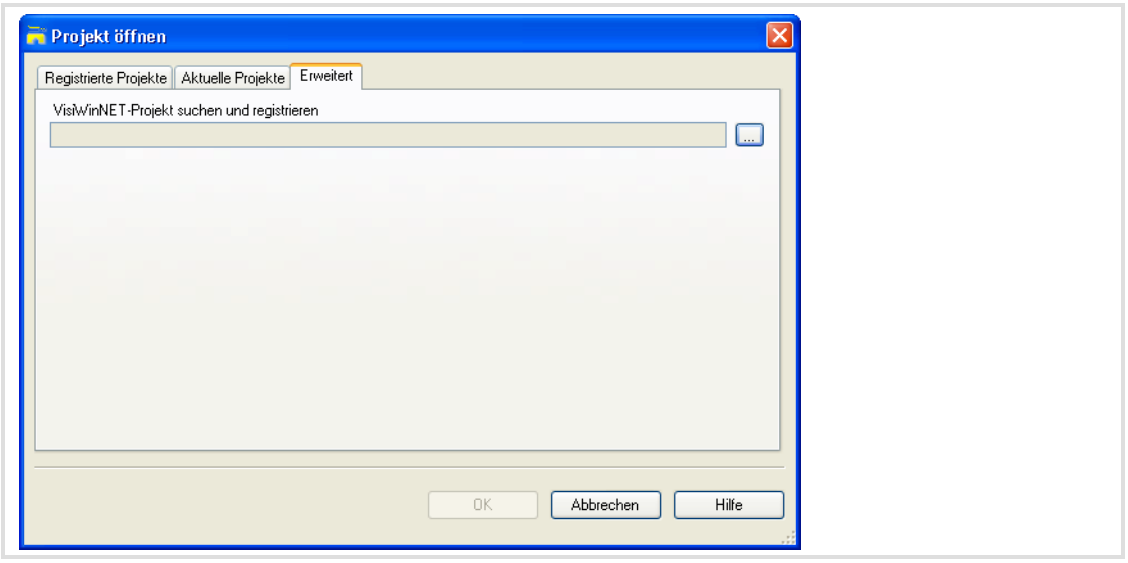
  Alle Informationen der Online-Hilfe finden Sie auch in den PDF-Dateien unter **Start** → **Alle Programme** → **Lenze** → **VisiWinNET 2005** → **Dokumentation x.x [PDF]**.

3.1.4 Projekt öffnen

Projekt registrieren und öffnen

So gehen Sie vor, wenn Sie ein Projekt öffnen möchten, das an Ihrem Arbeitsplatz noch nicht registriert ist (d. h. noch nie geöffnet wurde):

1. Klicken Sie im Menü **Datei** auf den Menübefehl **Projekt öffnen**.
   Der Dialog ”Projekt öffnen” wird angezeigt.
2. Wechseln Sie auf das Register **Erweitert**.

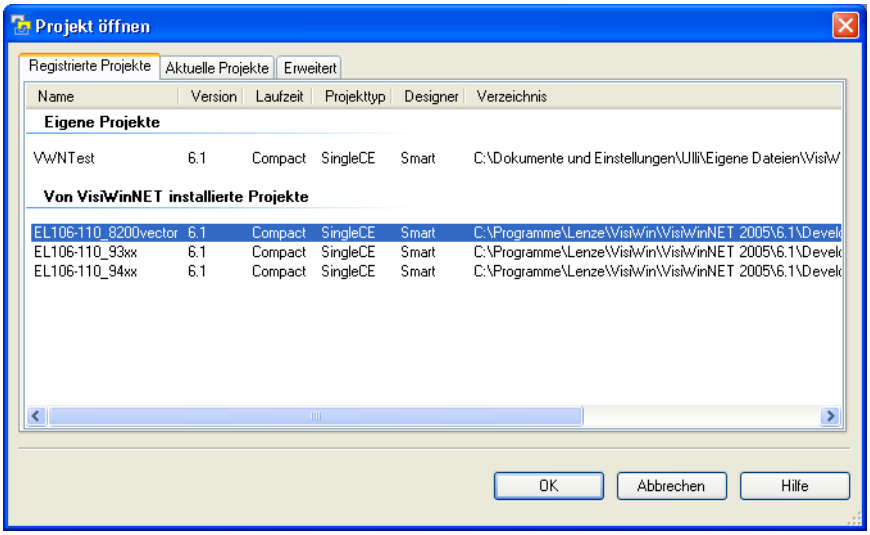


3. Klicken Sie auf die Schaltfläche **[...]**.
   Der Dialog ”Projekt öffnen” wird angezeigt.
4. Wählen Sie den gewünschten Projekt-Ordner.
   Standard: C:\Dokumente und Einstellungen\Benutzer\Eigene Dateien\VisiWinNET 2005
5. Wählen Sie dort die vwn-Datei und klicken Sie auf die Schalfläche **Öffnen**.
6. Schließen Sie den Dialog ”Projekt öffnen” über die Schaltfläche **OK**.
   Das Projekt wird geladen.

Registriertes Projekt öffnen

So gehen Sie vor, wenn Sie ein bereits registriertes Projekt öffnen möchten:

1. Klicken Sie im Menü **Datei** auf den Menübefehl **Projekt öffnen**.
   Der Dialog "Projekt öffnen" wird angezeigt.



2. Klicken Sie auf dem Register **Registrierte Projekte** auf das Projekt, das Sie öffnen möchten.
   Das Projekt wird geladen und im Projektexplorer angezeigt.

3.1.5 Projekt anlegen

Wie Sie ein neues Projekt in VisiWinNET® Smart anlegen können, erklären wir im Kapitel "Beispielprojekt" (📖 33).

3.1.6 Projekt speichern

Speichern

Eingaben im Clientbereich können Sie über den Menüpunkt **Datei** → **Speichern** (Eingaben in der aktuellen Ansicht) oder **Datei** → **Alles Speichern** (alle Eingaben) speichern.

Hinweis!

Sobald Sie versuchen eine noch nicht gespeicherte Ansicht bzw. ein noch nicht gespeichertes Projekt zu schließen, werden Sie durch einen Dialog darauf aufmerksam gemacht. Sie können dann die Änderungen annehmen (Schaltfläche **Ja**) oder ablehnen (Schaltfläche **Nein**).

Speichern unter einem anderen Namen

Wenn Sie das aktuelle Projekt unter einem neuen Namen speichern möchten, wählen Sie **Datei** → **Projekt speichern unter**. Die Änderungen am ursprünglichen Projekt werden nur im neuen Projekt gespeichert, das ursprüngliche bleibt unverändert.

Speichern als Projektvorlage

Über **Datei** → **Projekt speichern als Vorlage** können Sie eine Vorlage erstellen, auf deren Basis Sie bei der Erstellung von neuen Projekten aufsetzen können. Dies ist besonders dann zu empfehlen, wenn Sie öfters ähnliche Projekte erstellen und nicht immer ganz von vorne beginnen möchten.

Die Verwaltung von Projektvorlagen erlaubt dem Anwender das Abrufen eigener Applikationsrümpfe. Projektvorlagen dürfen sowohl fertig projektierte Formulare als auch Definitionen enthalten.

Hinweis!

Ggf. wird im Dialog ”Zusätzliche Dateien auswählen” gefragt, ob Dateien im Projektverzeichnis, die aus dem Projekt heraus nicht referenziert werden, in das neu erstellte Projektverzeichnis kopiert werden sollen. Gemeint sind hier insbesondere Dateien, wie z. B. HTML, XML, RTF oder Datenbanken, die von der Applikation in Steuerelementen verwendet werden, die aber nichts mit dem typischen Aufbau einer VisiWinNET SMART-Applikation zu tun haben.

3.1.7 Projekt konvertieren

Ein Projekt, das in einer älteren Programmversion erstellt wurde, konvertieren Sie, damit Sie darin alle erweiterten Funktionen der neuen Programmversion nutzen können.

Jede VisiWinNET® Smart-Version beinhaltet einen festen Funktionsumfang, der in den Projekten, die unter der Version erstellt wurden, zur Verfügung stehen.

Durch die Installation einer neuen VisiWinNET-Version gewinnt der Projekteur in der Regel neue Möglichkeiten. Bestehende Projekte können aber weiter mit dem alten Funktionsumfang geöffnet und bearbeitet werden.

Hinweis!

- Wenn Sie Projekte aus einer älteren Programmversion in eine neuere konvertieren möchten, müssen beide Programmversionen installiert sein.
- Ein Projekt, das in einer älteren Programmversion erstellt wurde, können Sie in einer neueren Programmversion ohne Konvertierung bearbeiten. Die erweiterten Programmfunktionen der neuen Programmversion stehen Ihnen dann aber nicht zur Verfügung.
- Das Konvertieren erspart Ihnen die Arbeit, ein Projekt neu aufbauen zu müssen, wenn sie auf eine neue Version umsteigen. Es entbindet sie jedoch nicht von der Pflicht, Ihre Applikation nach dem Konvertieren grundsätzlich noch einmal auf den Prüfstand zu stellen. Eine 100%ige Kompatibilitätsgarantie zwischen den Versionen gibt Lenze nicht.

  Eine Konvertierung beinhaltet automatisch auch das Risiko einer Verhaltensänderung Ihrer Applikation!

1. Starten Sie die neue Version von VisiWinNET® Smart.
2. Öffnen Sie das Projekt, das in einer älteren Programmversion erstellt wurde wie Eingangs beschrieben.
3. Klicken Sie im Menü **Datei** auf **Projekt konvertieren** und folgen Sie den Anweisungen.

   Ein neues Projekt wird angelegt. Das alte Projekt bleibt unverändert.

3.1.8 Programm beenden

So gehen Sie vor:

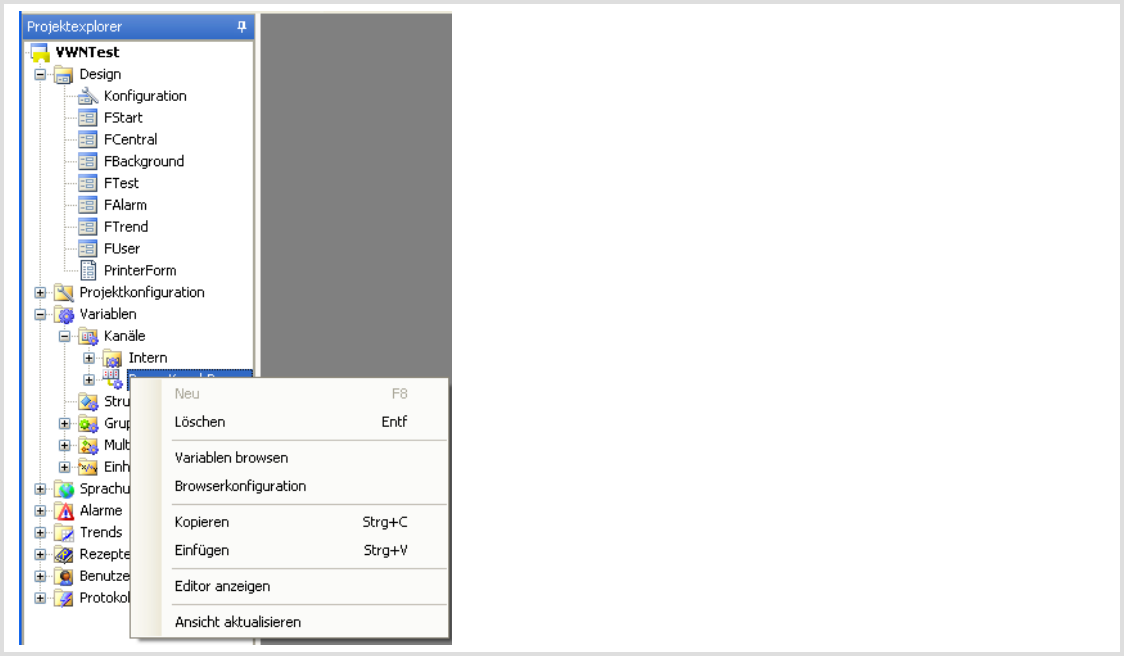
1. Klicken Sie im Menü **Datei** auf **Beenden**.

   Falls Sie Änderungen am Projekt vorgenommen haben und diese noch nicht gespeichert sind, erscheint ein Dialog, in dem alle Änderungen aufgelistet sind. Sie können diese Änderungen annehmen (Schaltfläche **Ja**) oder ablehnen (Schaltfläche **Nein**).

   VisiWinNET® Smart wird beendet.

3.2 Entwicklungsumgebung

3.2.1 Projektexplorer



Der Projektexplorer ermöglicht den Zugriff auf die Informationen der Projektdatenbank und die Formulare des geöffneten VisiWinNET® Smart-Projektes. Nach dem Start der Entwicklungsumgebung repräsentiert jeder Knoten in der Baumansicht des Projektexplorers ein System. Zusätzlich listet der Projektexplorer die Formulare des Projektes auf.

Die Bedienung des Projektexplorers ist vergleichbar mit der des Windows-Explorer.

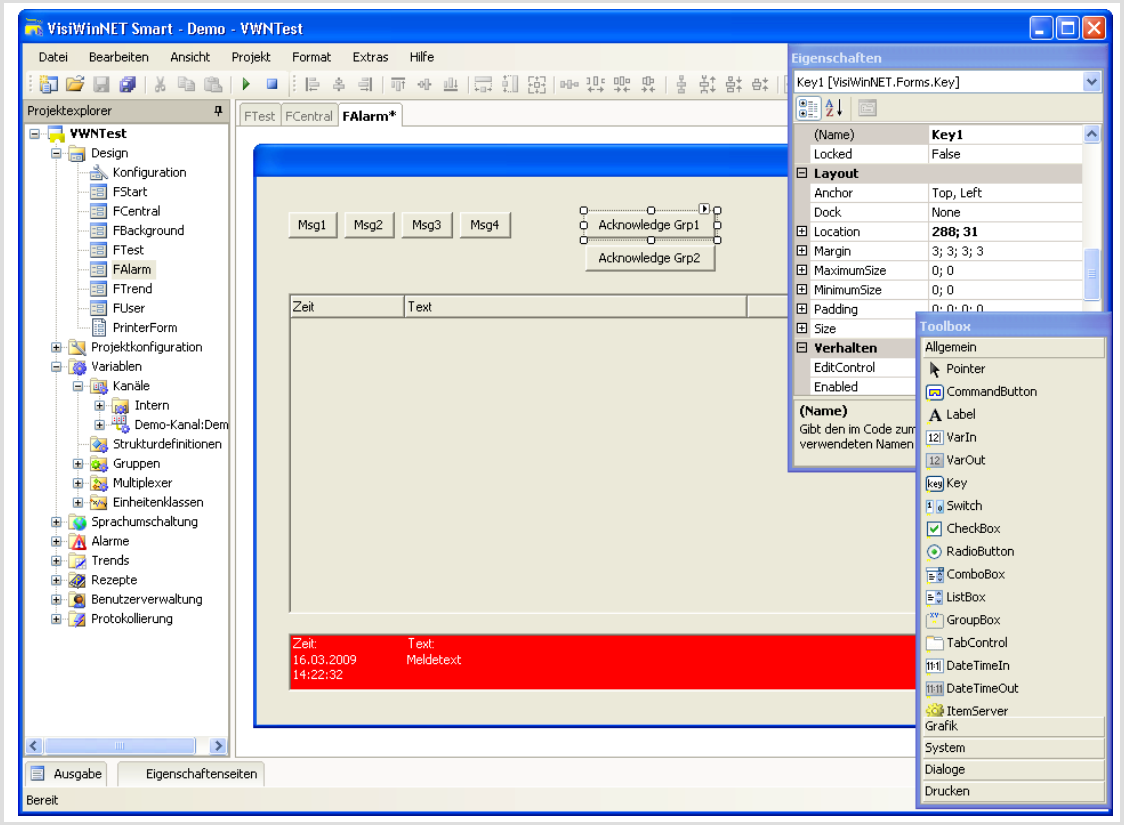
Kontextmenü

Jedes System enthält spezifische Funktionen. Diese Funktionen sind über das Kontextmenü des entsprechenden Knoten im Projektexplorer erreichbar. Das Kontextmenü wird über einen Klick mit der rechten Maustaste auf den entsprechenden Knoten geöffnet.

Beispiel für Kontextmenüs:

Knoten	Befehle des Kontextmenüs	Funktion
Design	Formular hinzufügen	Erzeugt ein leeres Formular. Dabei werden Formularabmessungen aus den Projekteinstellungen verwendet.
	Abgeleitetes Formular hinzufügen	Erzeugt ein neues Formular, das von einem Basisformular Eigenschaften und Steuerelemente erbt. Das Basisformular muss dabei in VisiWinNET Professional erzeugt, von "VisiWinNET.Forms.SmartForm" erben und in das Projekt kompiliert sein.
	Druckerformular hinzufügen	Erzeugt ein neues Druckformular
	Vorlage hinzufügen	Fügt ein bestehendes Formular aus der Vorlagensammlung ein.
	Vorhandene Datei hinzufügen	Fügt ein bestehendes Formular aus einem anderen Projekt ein.
	Ordner hinzufügen	Erzeugt im Projektverzeichnis einen Unterordner
	Einfügen	Fügt das Formular aus der Zwischenablage unter einem neuen Namen ein.
Formular	Öffnen	Öffnet die Designansicht des gewählten Formulars
	Aus Projekt entfernen	Entfernt den Verweis auf das Formular aus dem Projekt. Die Datei bleibt dabei erhalten.
	Kopieren	Kopiert das Formular in die Zwischenablage. Es kann über das Kontextmenü "Einfügen" des "Design"-Knotens unter einem neuen Namen in das Projekt wieder eingefügt werden.
	Löschen	Löscht das Formular aus dem Projekt. Löscht dabei auch die Formulardatei.
	Umbenennen	Ändert den Namen der Formulardatei
	Eigenschaften	Zeigt die Eigenschaften des Formulars im Eigenschaftenfenster.

3.2.2 Formulardesigner, Toolbox und Eigenschaftenfenster



Der Formulardesigner dient zum Erstellen bzw. Bearbeiten von:

- ► Bildschirmansichten für das Laufzeitsystem (📖 41).
- ► Druckformulare für die Ausgabe von Informationen während der Laufzeit (📖 88).

Er wird sichtbar, wenn Sie im Projektexplorer auf einen Formulareintrag doppelt klicken oder wenn Sie aus dem Kontextmenü des Formulareintrag den Eintrag ”Öffnen” wählen.

Beim Anlegen eines Projekts werden automatisch folgende drei Formulare angelegt:

FStart: Dieses Formular initialisiert das Laufzeitsystem. Es darf auf keinen Falle gelöscht werden!

FBackground: (Nur in Compact XP vorhanden.) Dieses Formular verdunkelt während der Laufzeit den Desktop-Hintergrund, wenn Sie auf Ihrem PC Applikationen mit einer kleineren Auflösung testen.

FCentral: Dieses Formular ist die zentrale Bildschirmansicht Ihres Laufzeitssystems. Von diesem Formular aus navigieren Sie zu weiteren Formularen und beenden die Applikation.

Toolbox

Die Toolbox enthält die Steuerelemente, die auf dem Formular platziert werden können. Dazu ist zuerst das gewünschte Steuerelement in der Toolbox zu markieren. Anschließend ist auf dem Formular die Stelle durch einen Klick mit der linken Maustaste zu wählen, wo das Steuerelement zu platzieren ist. Wird die Maustaste sofort wieder losgelassen, wird das Steuerelement an dieser Stelle mit einer vorgegebenen Größe platziert. Wird hingegen bei gehaltener Maustaste der Mauszeiger bewegt, bildet die Bewegung auf dem Bildschirm einen Rahmen. Das Loslassen der Maustaste bewirkt, dass das Steuerelement in der Größe des dargestellten Rahmens auf dem Formular erzeugt wird.

Position und Größe von Steuerelementen können nach dem Platzieren geändert werden. Wird ein Steuerelement mit einem Mausklick selektiert, wird um das Steuerelement herum ein Rahmen mit Ziehpunkten sichtbar.

Wie Sie ein Steuerelement verschieben:·

- ► Steuerelement mit Mausklick anwählen.
- ► Linke Maustaste gedrückt halten und Steuerelement auf die gewünschte Position verschieben.
- ► Maustaste loslassen

Wie Sie die Größe eines Steuerelementes ändern:

- ► Steuerelement mit Mausklick anwählen.
- ► Linke Maustaste auf einem Ziehpunkt an den Rändern gedrückt halten und Steuerelement auf die gewünschte Größe ziehen
- ► Maustaste loslassen

Hinweis!

Sie können auch mehrere Steuerelemente gleichzeitig verschieben oder skalieren, indem Sie diese nacheinander mit gedrückter [Strg]-Taste anwählen.

Typische Hilfsfunktionen zum Ausrichten oder Positionieren von Steuerelementen sind über die ”Format”-Werkzeugleiste aufrufbar.

Eigenschaftenfenster

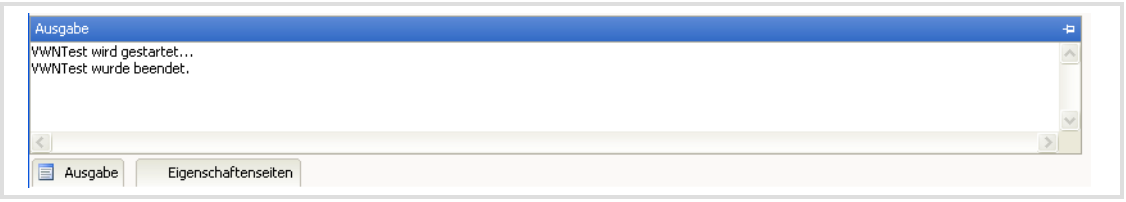
Die Eigenschaften eines Steuerelements geben Sie im Eigenschaftenfenster vor.

Hier legen Sie fest, ”was” das Steuerelement darstellen soll und ”wie” der Inhalt aussehen soll. Ein typisches Beispiel für das ”was” ist die Festlegung der darzustellenden Prozessgröße über die Eigenschaft ”VWItem”. Ein Beispiel für das ”wie” ist beispielsweise die Einstellung von Farben über ”ForeColor” und ”BackColor”.

Das Eigenschaftenfenster gibt immer die Eigenschaften des selektierten Steuerelementes wieder. Ein Steuerelement wird, wie oben beschrieben, durch einen Mausklick selektiert. Das selektierte Steuerelement wird durch einen Rahmen hervorgehoben.

Ist kein Steuerelement selektiert, werden die Eigenschaften des Formulars selbst angezeigt.

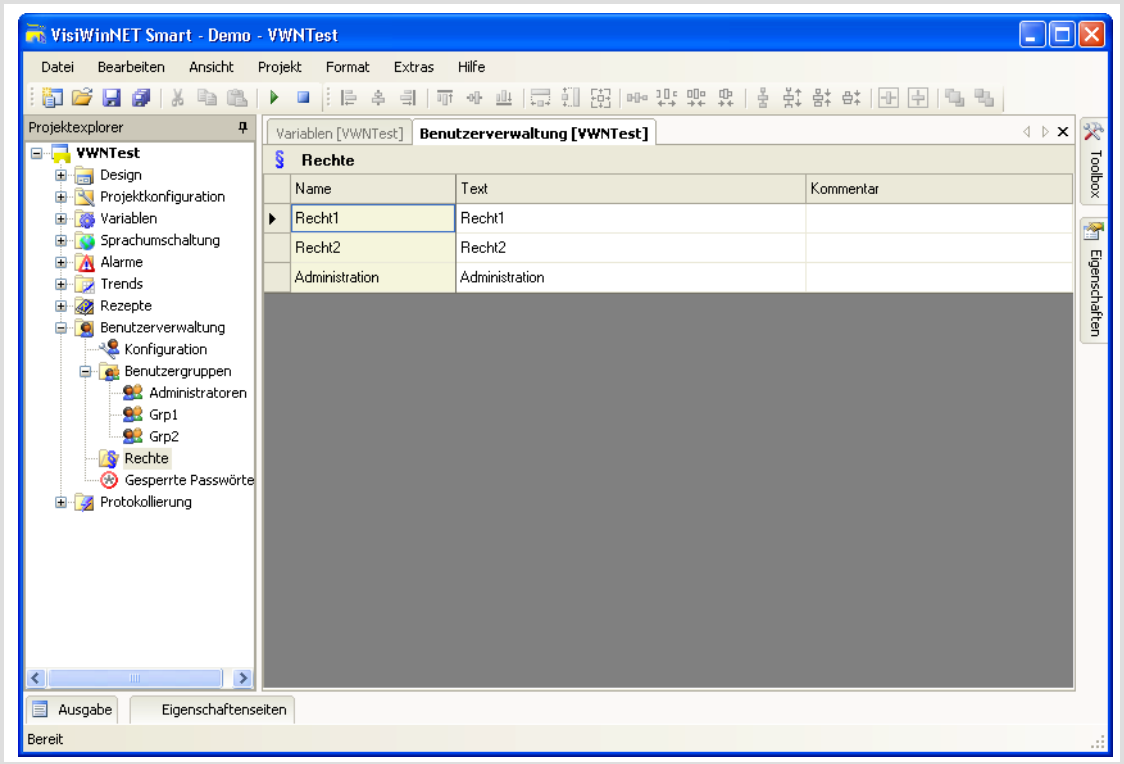
3.2.3 Ausgabefenster



Das Ausgabefenster enthält textuelle Informationen über:

- Ausnahmen, die zur Entwicklungszeit aufgetreten sind
- Den Fortschritt von Transferfunktionen zum Zielgerät

3.2.4 Tabelleneditor



Die Tabelleneditoren ermöglichen zusammen mit dem Projektexplorer den Zugriff auf die Definitionen der Projektdatenbank. Hier werden visualisierungsspezifische Definitionen vorgenommen, die zur Laufzeit den Zugriff auf Prozessdaten erlauben oder Sonderfunktionen, wie Alarme, Trends oder die Sprachumschaltung beschreiben.

Jedes System von VisiWinNET® Smart wird durch einen Tabelleneditor und Einträge im Projektexplorer repräsentiert. Ein Klick auf einen Eintrag im Projektexplorer öffnet den zum Eintrag zugehörigen Tabelleneditor. Die Tabelleneinträge repräsentieren die Definitionen der Projektdatenbank, die unter dem im Projektexplorer markierten Knoten gespeichert sind.

Jede Zeile ist eine Definition. Sie enthält die Werte der für die Definition typischen Parameter. Wenn Sie eine Definition anwählen, wird automatisch die zugehörige Eigenschaftenseite angezeigt. Über das Kontextmenü können neue Definitionen angelegt oder bestehende Definitionen geändert bzw. gelöscht werden.

Hinweis!

Wenn Sie auf einer Definition mit der rechten Maustaste klicken, wird das Kontextmenü geöffnet, über das Sie z. B. neue Definitionen anlegen können, bestehende duplizieren, löschen usw.

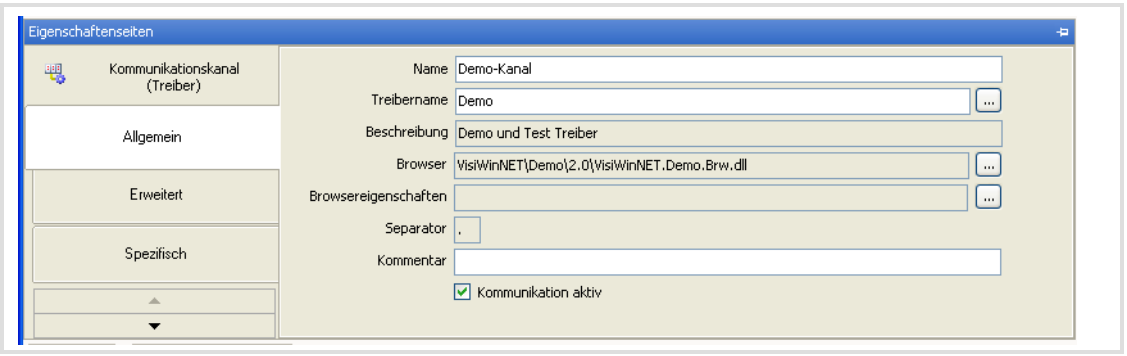
Über den Kontextmenübefehl ”Optionen” können Sie die Darstellung der Tabellen ändern. Sie können z. B. die angezeigten Spalten vorgeben oder die Reihenfolge der Spalten in der Tabelle.

Ein Teil der Definitionen wird vom Projektexplorer selbst verwaltet. Die vom Projektexplorer verwalteten Definitionen sind üblicherweise Gruppen, in denen sich weitere Definitionen befinden können. Ein typisches Beispiel hierfür sind die Gruppen des Alarmsystems, die im Projektexplorer als ein Zweig dargestellt werden. In diesen Gruppen können Untergruppen und Alarme projektiert werden. Angelegte Untergruppen werden im Projektexplorer dargestellt, weil auch sie wieder Definitionen enthalten können. Alarme können keine weiteren Definitionen enthalten. Daher werden sie im Tabelleneditor dargestellt.

Nicht alle Knoten im Explorer ermöglichen das Anlegen von Definitionen im Tabelleneditor. Beispielsweise können unter dem Knoten ”Archive” keine Trenddefinitionen angelegt werden. Trenddefinitionen können nur in einer übergeordnete Gruppe eine Funktion erfüllen. Diese Gruppe wäre in diesem Fall eine Archivdefinition. Zuerst ist also im Projektexplorer eine Archivdefinition anzulegen. Diese ist anschließend im Projektexplorer zu selektieren. Erst dann wird im Tabelleneditor das Anlegen von Trenddefinitionen möglich.

Teilweise sind bestimmte Gruppen auch gesperrt. Sie sind bereits in einem neuen Projekt angelegt und können nicht gelöscht werden. Der Inhalt dieser Gruppen kann auch nicht im Tabelleneditor erweitert werden. Diese Gruppen enthalten Definitionen, die vom Laufzeitsystem oder von den Steuerelementen erwartet werden. Definitionen in diesen Gruppen können aber parametriert werden.

3.2.5 Eigenschaftenseite



Die Eigenschaftenseite zeigt die Parameter der im Projektexplorer oder Tabelleneditor markierten Projektdatenbankdefinitionen.

Die Parameter sind je nach Definitionstyp über Karteikarten in unterschiedliche Kategorien unterteilt. Zwischen den Kategorien wird über die Karteireiter im linken Bereich der Eigenschaftenseite gewechselt.

Ein geänderter Wert wird übernommen, wenn die Eingabemarke das Fenster verlässt (z. B. wenn eine andere Definition markiert wird). Erst dann werden die geänderten Werte im Projektexplorer oder dem Tabelleneditor sichtbar.

3.2.6 Menüleiste

Die Einträge des Hauptmenüs beinhalten allgemeine Funktionen für die Entwicklungsumgebung und den Entwurf von Formularen. Weitere Funktionen von VisiWinNET® Smart finden Sie im Projektexplorer und den Tabelleneditoren der einzelnen Systeme.

Menü	Menübefehl	Funktion	Siehe auch
Datei		Projekt- und Modulverwaltungsaufgaben	
	Projekt Neu	Erzeugt ein neues VisiWinNET®Smart-Projekt.	📖 33
	Projekt Öffnen	Öffnet ein VisiWinNET®Smart-Projekt.	📖 12
	Schließen	Schließt die im Clientbereich geöffnete Ansicht. Ggf. werden Sie dazu aufgefordert, Änderungen zu speichern.	
	Projekt schließen	Schließt das aktuell in der Entwicklungsumgebung geöffnete Projekt. Ggf. werden Sie dazu aufgefordert, Änderungen zu speichern.	
	Speichern	Speichert die Änderungen in einem geöffneten Formular.	📖 14
	Projekt speichern unter	Speichert das Projekt unter einem neuen Namen in ein beliebiges Verzeichnis.	
	Projekt als Vorlage speichern	Speichert das Projekt als benutzerdefinierte Vorlage. Wenn Sie ein neues Projekt anlegen, steht Ihnen dann dieses Projekt als benutzerdefinierte Vorlage zur Verfügung.	
	Alles speichern	Speichert alle geänderten Formulare und Projektdateien.	
	Projekt konvertieren	Konvertiert das gerade geladene Projekt auf eine neuere Version. **Hinweis:** Das Konvertieren erspart Ihnen die Arbeit, ein Projekt neu aufbauen zu müssen, wenn sie auf eine neue Version umsteigen. Es entbindet sie jedoch nicht von der Pflicht, Ihre Applikation nach dem Konvertieren grundsätzlich noch einmal auf den Prüfstand zu stellen. Eine 100%ige Kompatibilitätsgarantie zwischen den Versionen gibt Lenze nicht.	📖 15
	Beenden	Beendet VisiWinNET®Smart. Vor dem Beenden werden Sie ggf. aufgefordert, Änderungen an den Formularen des Projektes zu speichern.	📖 15
Bearbeiten		Allgemeine Funktionen zum Ändern von Definitionen und Design	
	Rückgängig	Letzte Änderung rückgängig machen.	
	Wiederholen	Letzte rückgängig gemachte Änderung wieder herstellen.	
	Ausschneiden	Schneidet selektierte Steuerelemente aus dem Formular aus und kopiert sie in die Zwischenablage.	
	Kopieren	Kopiert die selektierten Steuerelemente in die Zwischenablage.	
	Einfügen	Fügt Steuerelemente aus der Zwischenablage in das aktive Formular.	
	Löschen	Löscht die selektierten Steuerelemente.	

Menü	Menübefehl	Funktion	Siehe auch
	Suchen	Sucht innerhalb des aktiven Fensters nach einem Begriff. Die Suche wird über einen Dialog gestartet. Hier sind folgende Einstellungen möglich: **Suchen in Spalte:** Gibt an, in welcher Tabellenspalte nach dem Suchbegriff gesucht werden soll. **Suchbegriff:** Gibt die zu suchende Zeichenkette an. Der Suchbegriff wird gegebenenfalls auf die möglichen Eingabewerte der zu durchsuchenden Spalte eingegrenzt. **Groß- und Kleinschreibung beachten:** Beachtet bei der Suche die Groß- und Kleinschreibung. **Gesamte Spalte suchen:** Ist diese Option aktiv, werden nur Felder angezeigt, deren Inhalt komplett mit dem Suchbegriff übereinstimmen. Mit Klick auf **Suchen** wird der Suchvorgang gestartet.	
	Weitersuchen	Setzt die letzte Suche an der aktuellen Position fort.	
	Ersetzen	Ersetzt den angegeben Suchbegriff durch einen neuen Text.	
	Alles markieren	Markiert alle Definitionen oder Objekte im aktiven Fenster.	
Ansicht		Aktivierung von Fenstern und Werkzeugleisten	
	Ausgabe	Schaltet das Ausgabefenster sichtbar	16
	Eigenschaften	Schaltet das Eigenschaftenfenster sichtbar	
	Projektexplorer	Schaltet das Projektexplorerfenster sichtbar	
	Toolbox	Schaltet das Toolboxfenster sichtbar	
	Symbolleisten -> Format	Schaltet die "Format"-Werkzeugleiste im Werkzeugleistenbereich sichtbar oder unsichtbar. Die Sichtbarkeit wird im Menü durch ein Kontrollkästchen angezeigt.	
	Symbolleisten -> Standard	Schaltet die "Standard"-Werkzeugleiste im Werkzeugleistenbereich sichtbar oder unsichtbar. Die Sichtbarkeit wird im Menü durch ein Kontrollkästchen angezeigt.	
Projekt		Laufzeitsystem bedienen	
	Starten	Startet das Laufzeitprobjekt.	
	Beenden	Beendet die laufende Instanz des Laufzeitprojektes.	
	Erstellen	Erstellt ein Laufzeitprobjekt.	
	Auf Zielgerät übertragen	Nur unter Windows XP oder Windows XP Embedded: Überträgt das Laufzeitprobjekt auf ein Zielgerät, ohne es zu starten. Ggf. wird dabei ein Verbindungsdialog angezeigt, wenn noch keine Verbindung hergestellt wurde. (Siehe im Menü Extras "Zielgerät verbinden".)	94
	Auf Zielgerät starten	Startet das Laufzeitprobjekt auf einem Zielgerät. Ggf. wird dabei ein Verbindungsdialog angezeigt, wenn noch keine Verbindung hergestellt wurde. (Siehe im Menü Extras "Zielgerät verbinden".)	
Format		Das Menü "Format" beinhaltet Funktionen, die auf Steuerelemente in Formularen angewendet werden können, um sie z. B. zu skalieren oder zu positionieren.	
Extras		Verbindung zum Zielgerät	
	Zielgerät verbinden	Stellt die Verbindung zu einem "Windows CE"-Gerät her. Auf der Gegenseite wird der VisiWinNET®Smart-RemoteAccess-Manager erwartet.	
	Verbindung trennen	Trennt die Verbindung zum "Windows CE" Zielgerät. Dieses Menü steht nur bei "Compact"-Projekten für Windows-CE zur Verfügung.	

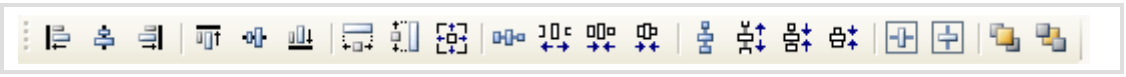
Menü	Menübefehl	Funktion	Siehe auch
	Optionen	Beinhaltet u. a. folgende Einstellungsmöglichkeiten: **Allgemein:** Gibt die Anzahl an Projekten vor, die beim ”Projekt öffnen” in der Aktuell-Liste angezeigt werden sollen. **Verzeichnis für eigene Projekte:** Angabe des Verzeichnisses, das beim Anlegen eines neuen Projektes als Vorgabe eingestellt ist. **Verzeichnis für eigene Vorlagen:** Verzeichnis, in dem eigene Vorlagen abgelegt werden. Aus diesem Verzeichnis sind Formulare als Vorlagen in anderen Projekten auswählbar.	
Hilfe		Aufruf von Hilfe- und Supportinformationen	
	Inhalt, Index oder Schnelleinstieg	Ruft die Onlinehilfe auf.	
	Netviewer Sitzung starten		
	Über ...	Ruft einen Dialog auf, der Versionsinformationen über die installierten VisiWinNET®Smart-Produkte enthält.	

3.2.7 Werkzeugleiste ”Standard”

Den Schaltflächen der Format-Werkzeugleiste sind häufig verwendete Befehle aus den Menüs **Datei**, **Bearbeiten** und **Projekt** zugeordnet. Wenn Sie den Mauszeiger auf ein Symbol positionieren, wird nach einer kurzen Zeit ein Tooltip mit dem Menübefehl eingeblendet.

Die ”Standard”-Werkzeugleiste wird über den Menübefehl **Ansicht** → **Symbolleisten** → **Standard** sichtbar geschaltet.

3.2.8 Werkzeugleiste ”Format”



Den Schaltflächen der Format-Werkzeugleiste sind häufig verwendete Befehle aus dem Menü **Format** zugeordnet. Wenn Sie den Mauszeiger auf ein Symbol positionieren, wird nach einer kurzen Zeit ein Tooltip mit dem Menübefehl eingeblendet.

Die Format-Werkzeugleiste wird über den Menübefehl **Ansicht** → **Symbolleisten** → **Format** sichtbar geschaltet.

3.3 Emulator für Windows® CE-Projekte

Ab der Version 6.5 beinhaltet **VisiWinNET® Smart** einen Emulator für Windows® CE-Projekte.

Wenn während der Entwicklung von Windows® CE-Projekten noch kein Zielgerät zur Verfügung steht, kann über diesen Emulator eine Visualisierung auf dem Entwicklungs-PC getestet werden.

Hinweis!

Bitte beachten Sie, dass es sich dabei nur um ein Vortest während der Entwicklungsphase handelt. Er ersetzt keinen finalen Test auf dem Zielsystem.

Funktionale Unterschiede zwischen Emulation und realem Laufzeitsystem:

- Datenanbindung

  Alle Kommunikationskomponenten (und damit die reale Verbindung zur SPS) werden simuliert. Über die Oberfläche des Emulators können während der Simulation Variablen gesetzt werden.

- Framework

  Der Emulator verwendet das Microsoft® .NET Standard Framework. Deshalb können funktionale Unterschiede auftreten, wenn Funktionen des Microsoft® .NET Compact Frameworks unter Windows® CE verwendet werden.

- Performance

  Ladezeiten oder zeitliche Abläufe des finalen Laufzeitsystems hängen stark von der Leistung des Zielgerätes ab.

3.3.1 Emulator starten

Hinweis!

Der Emulator kann nur aus einem geladenen Windows® CE-Projekt heraus gestartet werden.

Sie können den den Emulator mit folgenden Optionen starten:

- Emulator mit Variablensimulation
  Simuliert den Zugriff auf SPS-Variablen.
- Emulator mit Kommunikationstreiber
  Der Kommunikationstreiber baut eine reale Verbindung zur projektierten SPS auf. Er unterstützt folgende Lenze-Treiber:
  – LogicAndMotion V3
  – CoDeSys
  – S7 TCP/IP
  Sofern nicht unterstützte Kommunikationskomponenten projektiert sind, werden diese beim Start des ”Emulators (mit Kommunikationstreiber)” deaktiviert.

So gehen Sie vor:

1. Im Werkzeugleistenbereich das Kombinationsfeld öffen und dort die gewünschte Option wählen.

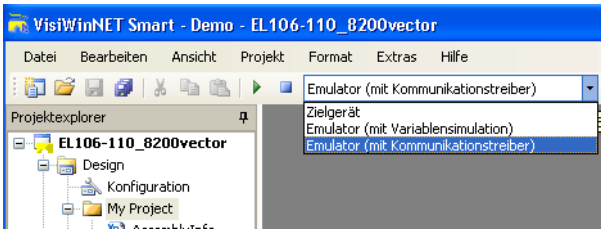


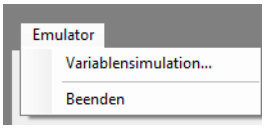
Abb. 3-1 Emulationsvariante wählen

2. Projekt wie gewohnt kompilieren und starten, z. B. mit [F5].
   Das Emulator-Fenster mit der Applikationsoberfläche wird geöffnet.

3.3.2 Emulator-Fenster

Abb. 3-2 Beispiel für ein Emulator-Fenster

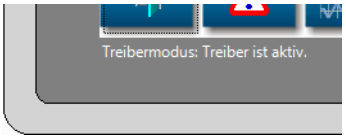
Emulator-Menü



Variablensimulation: Variablen-Simulationsfenster öffnen.

Beenden: Emulator beenden.

Treibermodus

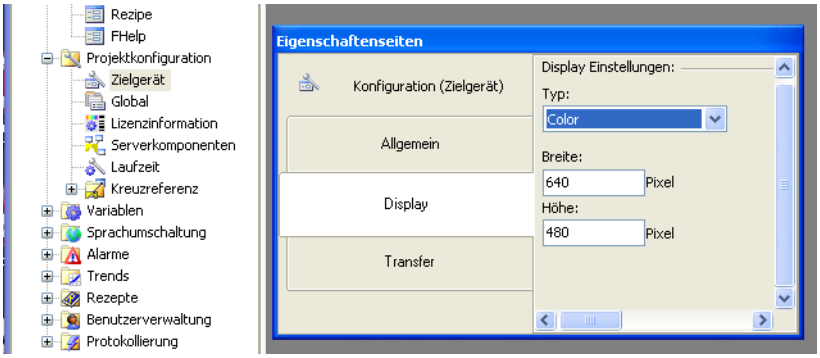


Treibermodus Simulation: Emulator mit Variablensimulation ist aktiv. Der Zugriff auf SPS-Variablen wird simuliert.

Treibermodus: Treiber ist aktiv: Emulator mit Kommunikationstreiber ist aktiv. Das Projekt baut im Emulator eine aktive Verbindung zur projektierten Steuerung auf. Die Variablenwerte werden also nicht simuliert, sondern stammen von der Steuerung.

Fenstergröße

Die Größe des Fensters wird im Objektbaum unter **Projektkonfiguration** → **Zielgerät** → Karteikarte **Display** vorgegeben.



Hinweis!

Das Emulationsfenster kann maximal die Auflösung des Zielsystems annehmen. Die Darstellung der Applikation wird abhängig von der Eigenschaft "ScaleToScreenSize" skaliert bzw. abgeschnitten. Sie kann dadurch von der Darstellung auf dem Zielgerät abweichen.

3.3.3 Variablen-Simulationsfenster

Aufruf: Im Menü des Emulator-Fensters auf ”Variablensimulation” klicken.

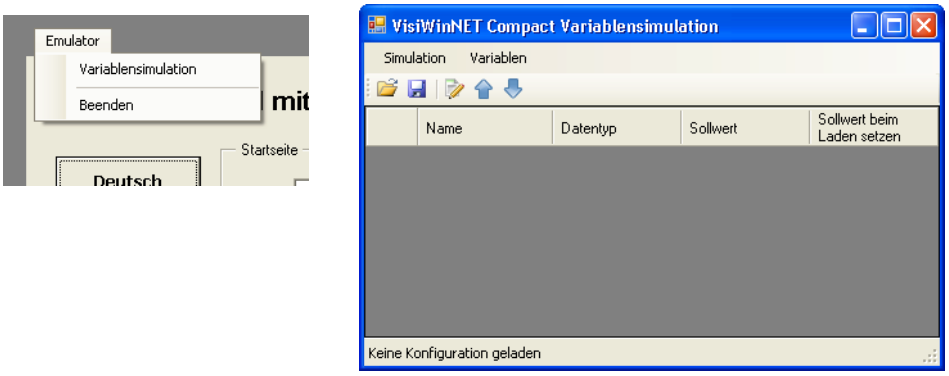
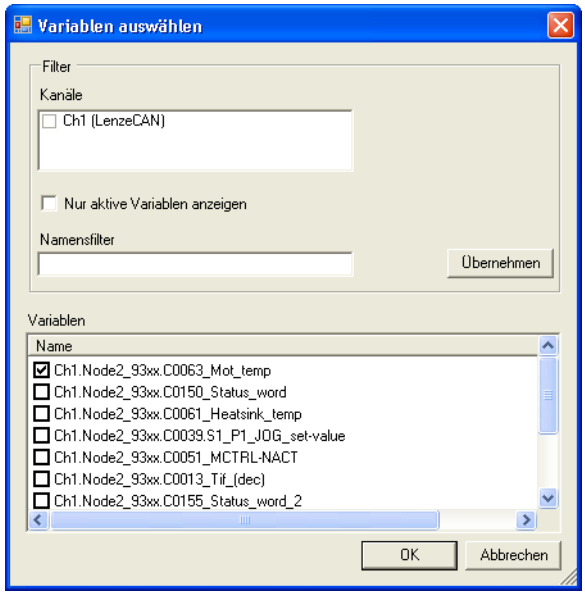


Abb. 3-3 Variablen-Simulationsfenster ohne geladene Konfiguration

Variablen auswählen

Aufruf: Im Variablen-Simulationsfenster auf **Variablen** → **Auswählen** klicken.



Bereich ”Filter”

Kanäle: Kanal wählen, dessen Variablen im unteren Fensterbereich angezeigt werden sollen.

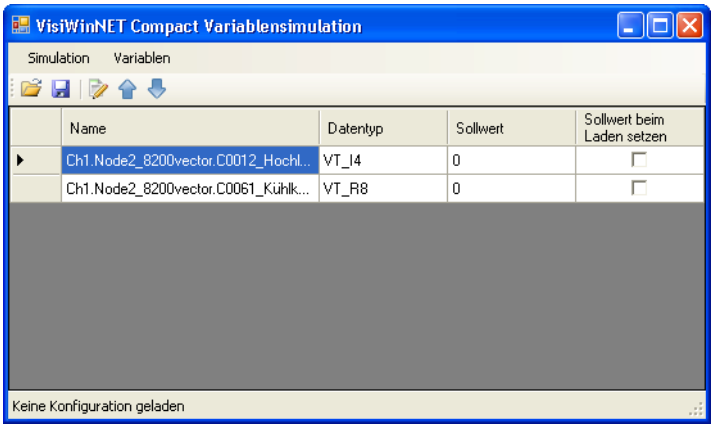
Nur aktive Variablen anzeigen: Wenn markiert, werden nur aktive Variablen im unteren Fensterbereich angezeigt.

Namensfilter: Filter für Variablenanzeige vorgeben. Groß-/Kleinschreibung wird unterschieden. Wildcards (”*”, ”?” usw.) sind nicht zulässig.

Variablen: Variablen wählen, deren Werte simuliert werden sollen.

[OK]: Fenster schließen. Die markierten Variablen werden im Listenfeld des Variablen-Simulationsfenster angezeigt.

Sollwerte vorgeben



Listenfeld

Name: Name und Adresse der Variablen

Datentyp: Typ der Variablen (nur Anzeige)

Sollwert: Gewünschter Wert der Variablen hier eingeben.

Sollwert beim Laden setzen: Wenn markiert, wird der Wert bei Laden einer Konfigurationsdatei direkt geschrieben.

Menü ”Variablen”

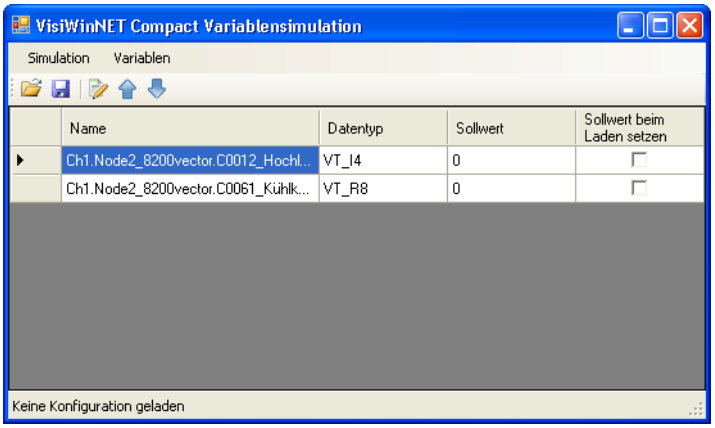
Auswählen: Fenster ”Variablen auswählen” öffnen

Sollwerte schreiben: Variablenwerte in den Variablenkern übertragen. Anschließend springt der aktuelle Wert um und zeigt den Sollwert.

Aktuelle Werte lesen: Variablenwerte aus dem Variablenkern lesen.

Konfigurationsdatei erstellen und laden

Die Liste, der im Variablen-Simulationsfenster angezeigten Variablen einschließlich deren Werte, kann in einer Datei gespeichert werden. Für spätere Emulationsläufen können die gespeicherten Variablenlisten geladen werden.



Menü ”Simulation

Konfiguration laden: Existierende Konfiguration laden. Die aktuell geladene Konfiguration wird im unteren Fensterbereich angezeigt.

Konfiguration speichern/speichern unter: Aktuelle Variablenauswahl als Konfiguration speichern.

Konfiguration löschen: Aktuell geladene Konfiguration löschen.

Initialkonfiguration festlegen: Aktuell geladene Konfiguration als Initialkonfiguration deklarieren.

Beim erneuten Start des Projekts im Emulator wird die Initialkonfiguration automatisch geladen. Alle Sollwerte, bei denen die Option ”Wert beim Laden schreiben” aktiv ist, werden direkt in den Variablenkern geschrieben. Sollwerte, bei denen die Option nicht aktiv ist, werden erst geschrieben, wenn die Funktion ”Sollwerte schreiben” aus dem Menü ”Variablen” aufgerufen wird.

Hinweis!

Wir empfehlen, eine Konfigurationsdatei für jeweils einen Funktionsblock (Formular oder Teilbereich eines Formulars) zu definieren.

4 Beispielprojekt

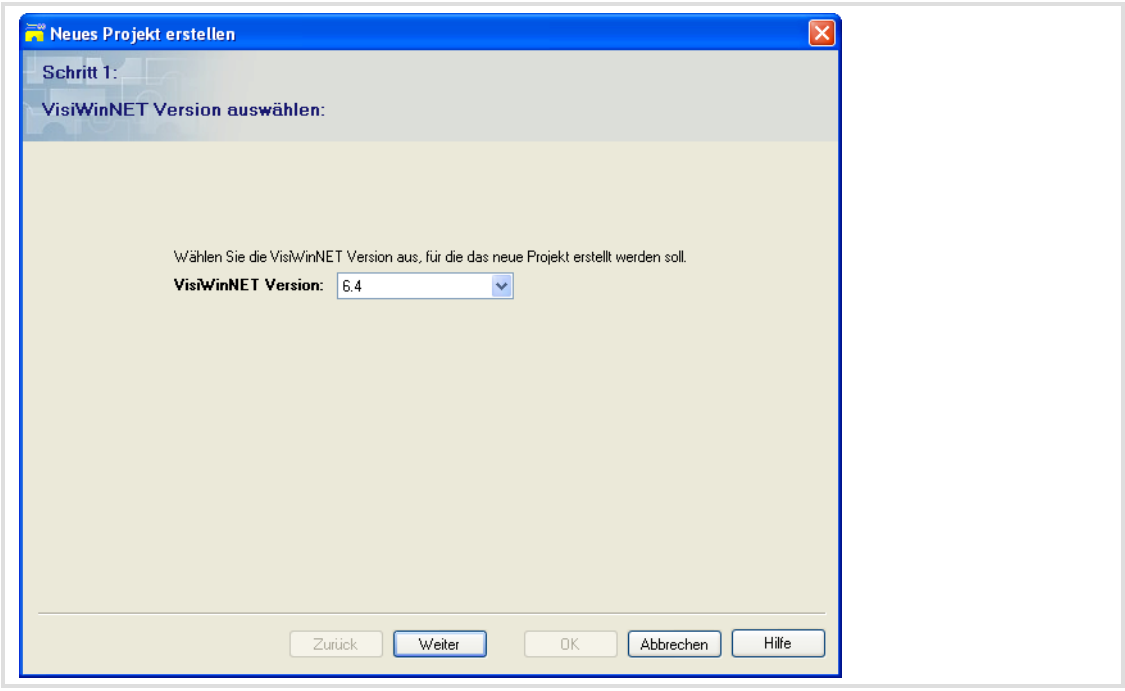
Dieses Kapitel beschreibt die grundlegenden Vorgehensweisen bei der Erstellung einer Visualisierungsapplikation anhand von Beispielen.

Wenn Sie die vorgegebenen Schritte nachvollziehen, lernen Sie die wesentlichen Funktionen von VisiWinNET® Smart schnell und effizient kennen.

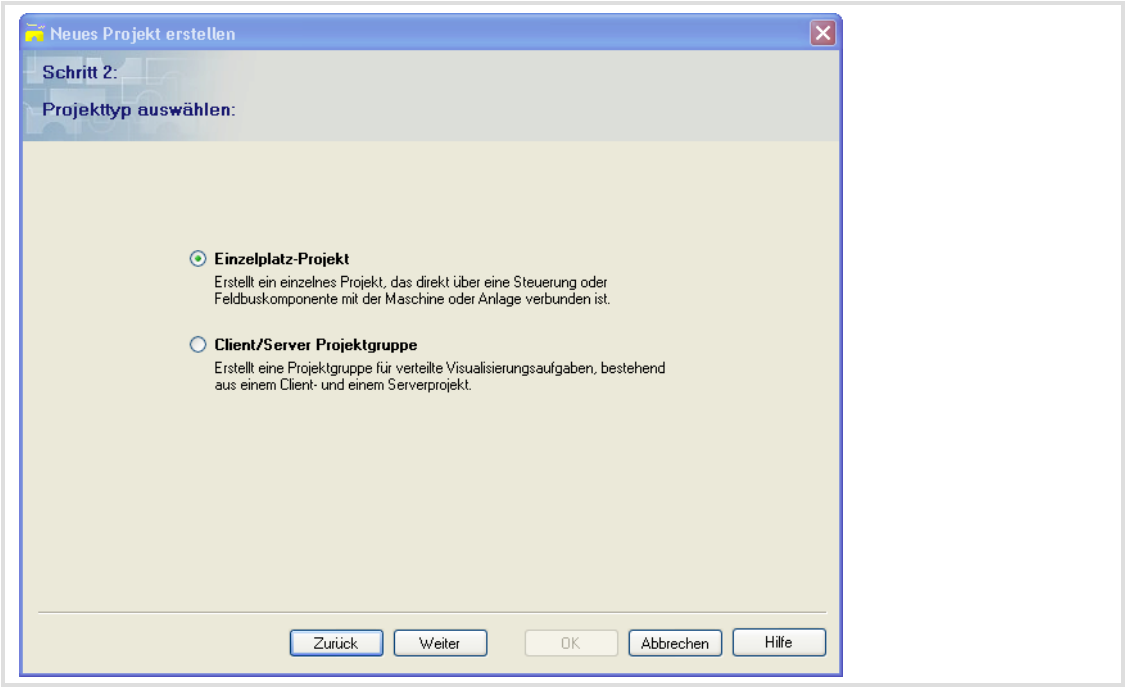
Das Beispiel basiert auf dem VisiWinNET® Smart-Demotreiber. Das heißt, Sie benötigen kein externes Zielsystem. Die im Folgenden beschriebenen Schritte können Sie also bequem auf Ihrem Desktop-Rechner oder Laptop nachvollziehen, wenn VisiWinNET® Smart dort installiert ist.

4.1 Beispielprojekt anlegen

1. Klicken Sie in VisiWinNET® Smart auf **Datei** → **Projekt Neu**.
   Im folgenden Dialog können Sie vorgeben, für welche Programmversion Sie das Projekt erstellen möchten (📖 15).

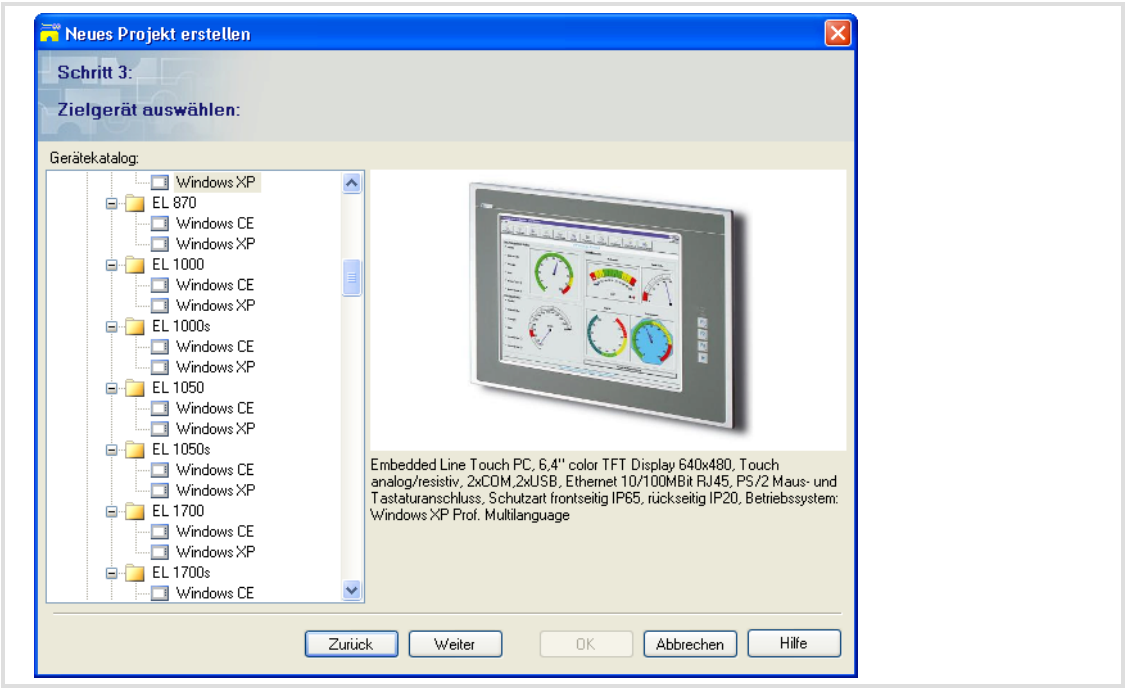


2. Klicken Sie auf **Weiter**.
   Im folgenden Dialog können Sie wählen, ob Sie ein Einzelplatz-Projekt oder ein Client-/Server-Projekt erzeugen möchten.

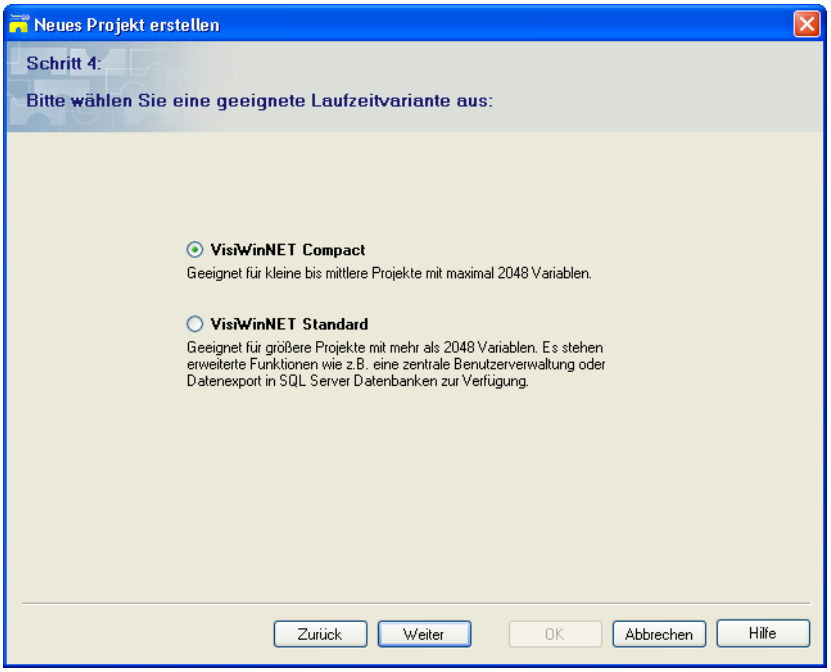


Wir beschreiben im Folgenden die Erstellung eines Einzelplatz-Projekts.

3. Markieren Sie **Einzelplatz-Projekt** und klicken Sie auf **Weiter**.
   Im folgenden Dialog wählen Sie das Zielgerät für dieses Projekt.



4. Da wir das Beispielprojekt mit dem Demotreiber aufbauen, können Sie einen beliebigen **IPC** wählen.
5. Wählen Sie das Betriebssystem **Windows XP** und klicken Sie auf **Weiter**.
   Im folgenden Dialog können Sie wählen, ob Sie ein ”Compact”- oder ein ”Standard”-Projekt erzeugen möchten.

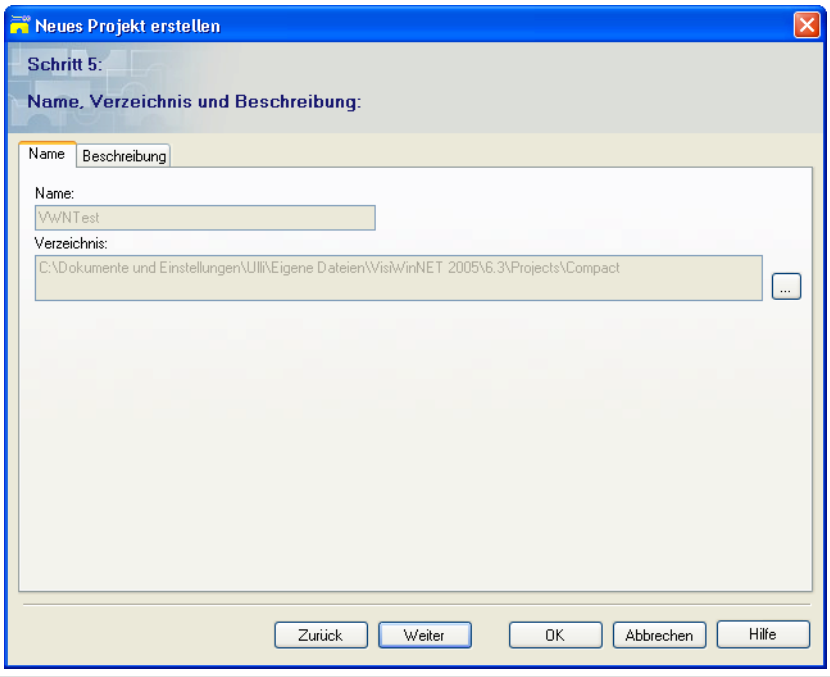

Neues Projekt erstellen
Schritt 4:
Bitte wählen Sie eine geeignete Laufzeitvariante aus:
VisiWinNET Compact
Geeignet für kleine bis mittlere Projekte mit maximal 2048 Variablen.
VisiWinNET Standard
Geeignet für größere Projekte mit mehr als 2048 Variablen. Es stehen erweiterte Funktionen wie z.B. eine zentrale Benutzerverwaltung oder Datenexport in SQL Server Datenbanken zur Verfügung.
Zurück
Weiter
OK
Abbrechen
Hilfe

Wir beschreiben im Folgenden die Erstellung eines ”Compact”-Projekts.

6. Markieren Sie **VisiWinNET Compact** und klicken Sie auf **Weiter**.

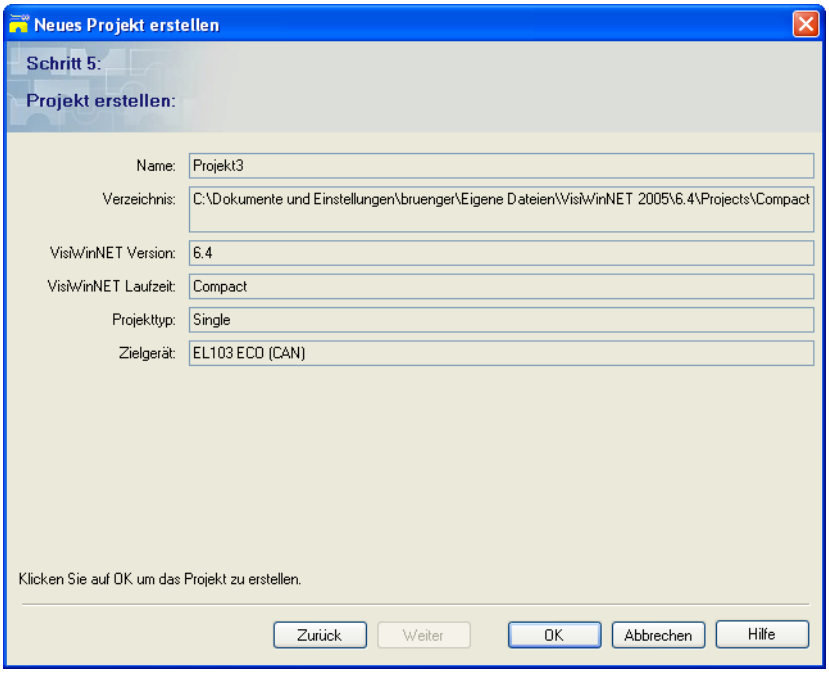
   Im folgenden Dialog können Sie einen Namen und ein Speicherort für das Projekt vorgeben.

   Im Demo-Modus (ohne Dongle) können Sie den vorgegebenen Projektnamen ”VWNTest” nicht ändern.



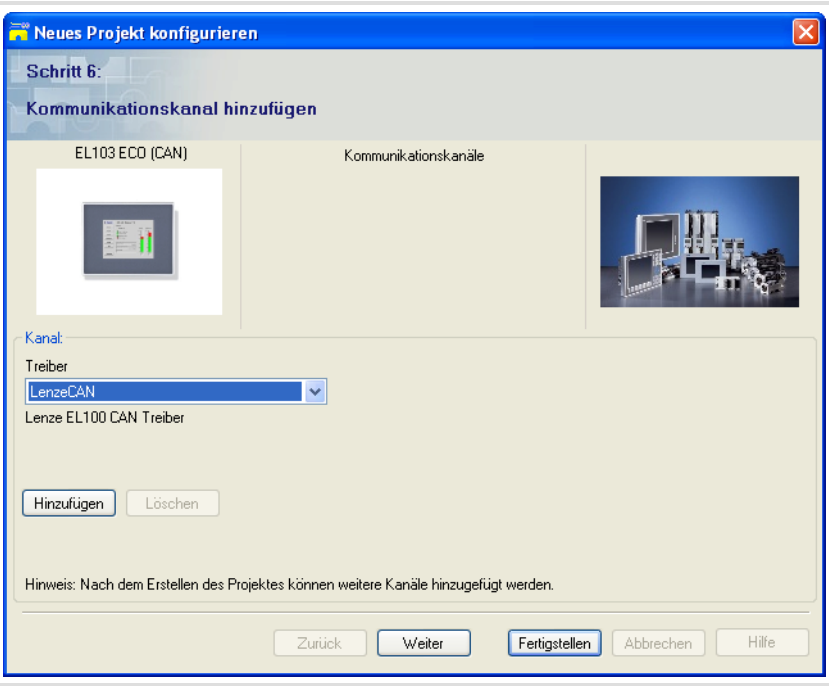
7. Klicken Sie auf **Weiter**.

   Der folgende Dialog zeigt eine Übersicht aller Projekt-Einstellungen.



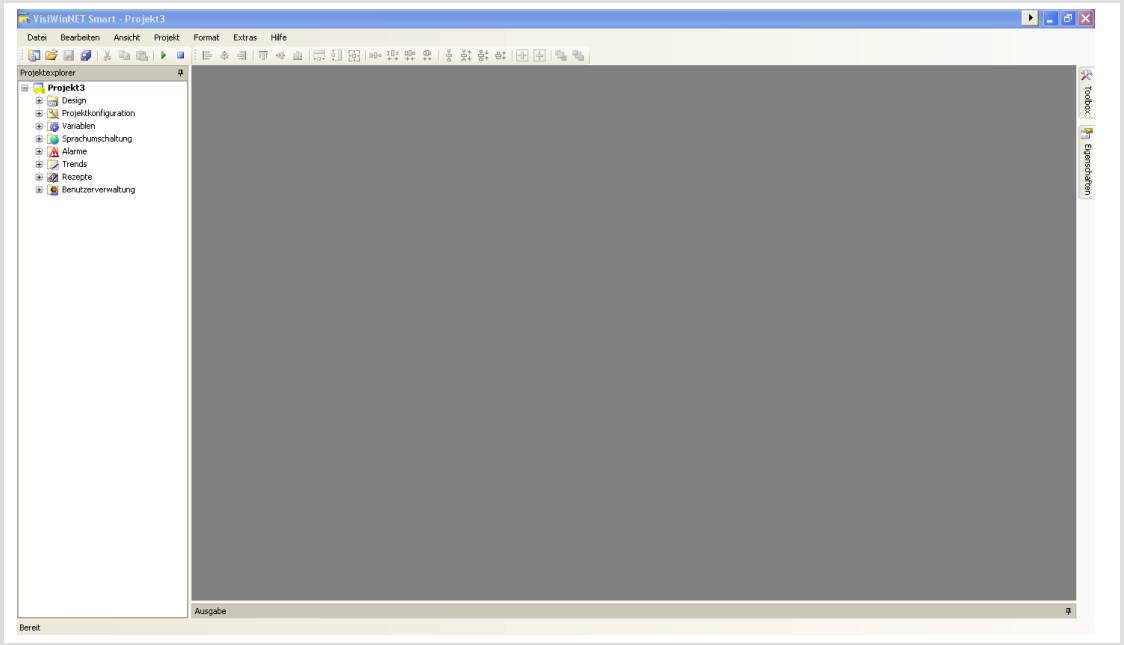
8. Klicken Sie auf **OK**, um das Projekt anzulegen.

   In den folgenden Dialogen können Sie einen Kommunikationskanal anlegen und konfigurieren sowie weitere Projektdaten vorgeben. Dies wollen wir an dieser Stelle zunächst überspringen. Wir beschreiben eine alternative Vorgehensweise im nächsten Abschnitt.



9. Klicken Sie auf **OK**.

   Das Projekt wird angelegt und im Projektexplorer angezeigt.


VisiWinNET Smart - Projekt3
Datei
Bearbeiten
Ansicht
Projekt
Format
Extras
Hilfe
Projektexplorer
Projekt3
Design
Projektkonfiguration
Variablen
Sprachumschaltung
Alarme
Trends
Rezepte
Benutzerverwaltung
Toolbox
Eigenschaften
Ausgabe
Bereit

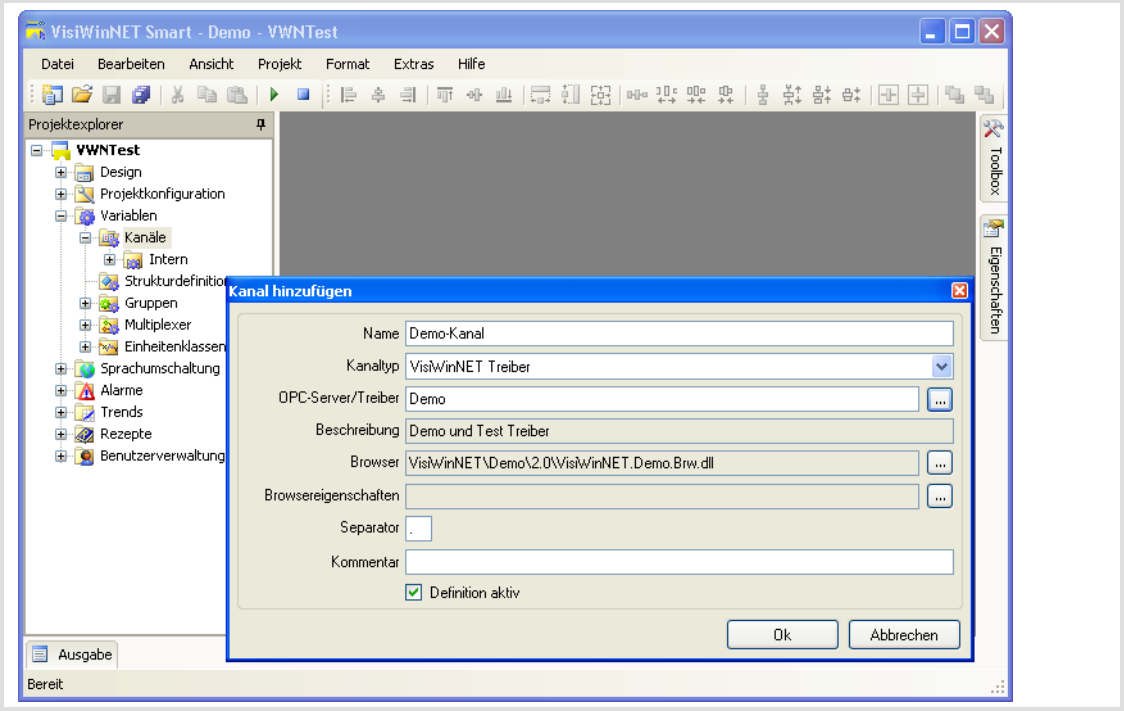
4.2 Kommunikationskanal erstellen

Die Definition eines Kommunikationskanal legt fest, mit welcher Kommunikationskomponente (z. B. HMI 100 oder IPC) Daten ausgetauscht werden sollen. Wir erstellen hier einen Kommunikationskanal mit dem Demo-Treiber.

1. Öffnen Sie im Projektexplorer den Knoten **Variablen** und markieren Sie dort den Knoten **Kanäle**.
2. Klicken Sie auf dem Knoten **Kanäle** mit der rechten Maustaste und wählen Sie im Kontextmenü den Eintrag **Neu**. Alternativ können Sie auch die Funktionstaste **[F8]** drücken.

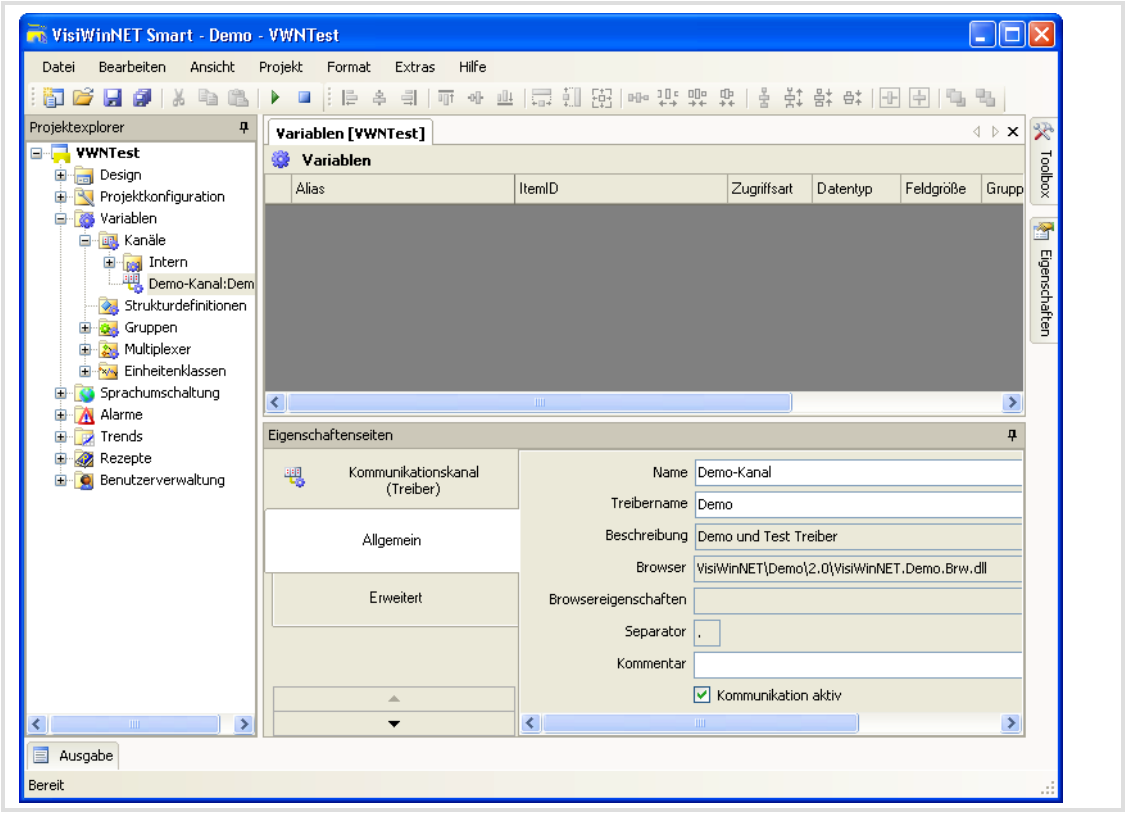
   Der Dialog ”Kanal hinzufügen” wird geöffnet.
3. Geben Sie als Name **Demo-Kanal** ein. Klicken Sie anschließend neben dem Feld ”OPC-Server/Treiber” auf die Schaltfläche **[...]** und wählen im Auswahldialog den **Demo-Treiber.**

   Nachdem Sie den Auswahldialog mit **OK** geschlossen haben, sollte der Dialog ”Kanal hinzufügen” wie folgt aussehen:



4. Klicken Sie auf **OK**.

   Der Kanal wird angelegt und im Projektexplorer angezeigt.



5. Um zu kontrollieren, ob der Kommunikationskanal korrekt funktioniert, können Sie im Menü **Projekt** auf den Menüpunkt **Starten** klicken.

   (Alternativen: **[F5]** drücken oder die Schaltfläche ▶ auf der Symbolleiste ”Standard”.)

   Wenn der Kommunikationskanal einwandfrei funktioniert, wird kurzzeitig eine VisiWinNET®-Grafik eingeblendet und anschließend ein Visualisierungsformular mit einer Schaltfläche ”Ende”.

Hinweis!

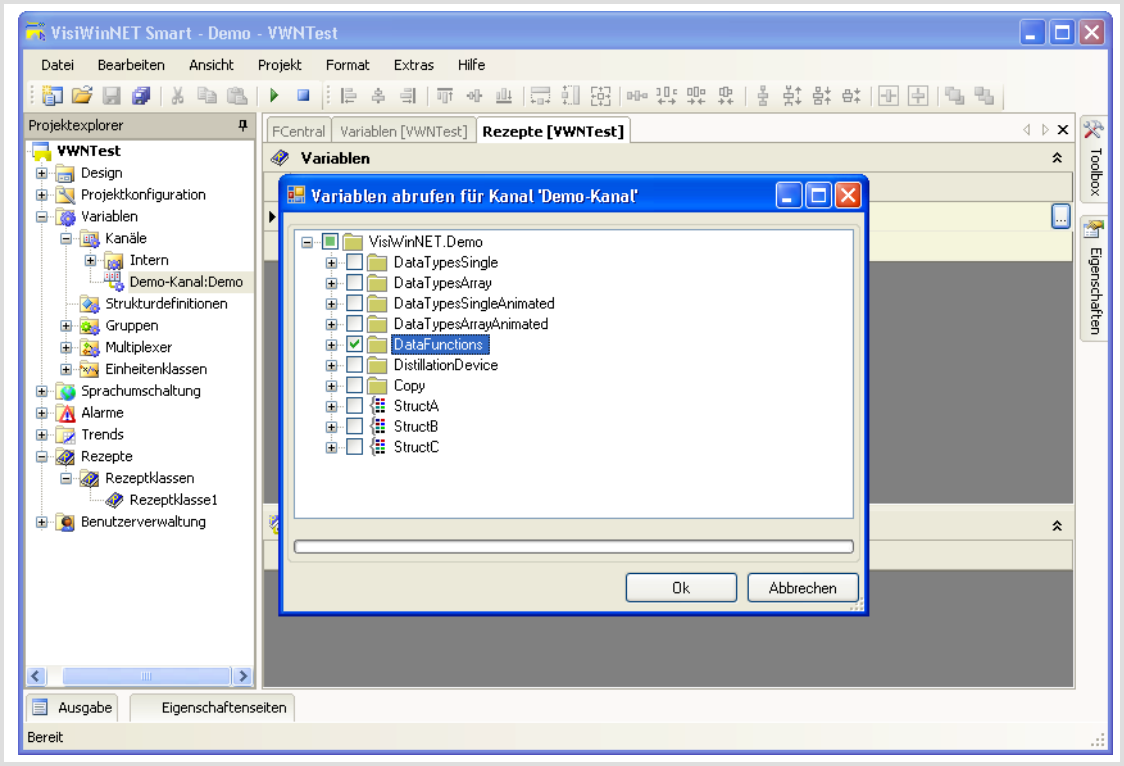
Wenn Sie im Demo-Moduls (ohne Dongle) arbeiten, wird bei jedem Start des Laufzeitsystems ein Hinweis-Dialog eingeblendet. Schließen Sie ihn über die Schaltfläche **OK**.

4.3 Variablen browsen

Dieser Abschnitt beschreibt das Einlesen von Variablendefinitionen aus Projektdateien, der SPS oder anderen Quellen. Wie wählen hier für unser Beispielprojekt eine Quelle, die uns Beispielwerte, wie zum Beispiel eine Sägezahnkurve, liefert. Weitere Beispiele für das Browsen von Variablen finden Sie im Kapitel ”5.2 Kommunikation”.

1. Öffnen Sie im Projektexplorer den Knoten **Variablen** → **Kanäle**.
2. Klicken Sie auf dem Objekt **Demo-Kanal** mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Variablen browsen**.

   Der Dialog ”Variablen abrufen für Kanal ...” wird geöffnet.



3. Markieren Sie das Kontrollfeld **DataFunction** und klicken Sie auf **OK**.

   Im Projektexplorer erscheint unterhalb des Objekts ”Demo-Kanal” der Eintrag ”Data-Function”.

4.4 Eine Visualisierung erstellen

Dieser Abschnitt beschreibt den allgemeinen Umgang mit Formularen (Designs) und dessen Steuerelementen.

Ein Formular repräsentiert eine Bildschirmansicht oder einen Dialog. Es visualisiert die Prozessvariablen aus der Projektdatenbank und enthält Steuerelemente z. B. für die Navigation zwischen weiteren Formularen.

4.4.1 Formular erstellen und Werte darstellen

Nach dem Anlegen eines neuen Projektes sind einige Formulare bereits angelegt.

FStart: Dieses Formular initialisiert das Laufzeitsystem. Es darf auf keinen Falle gelöscht werden!

FBackground: Dieses Formular verdunkelt während der Laufzeit den Desktop-Hintergrund, wenn Sie auf Ihrem PC Applikationen mit einer kleineren Auflösung testen.

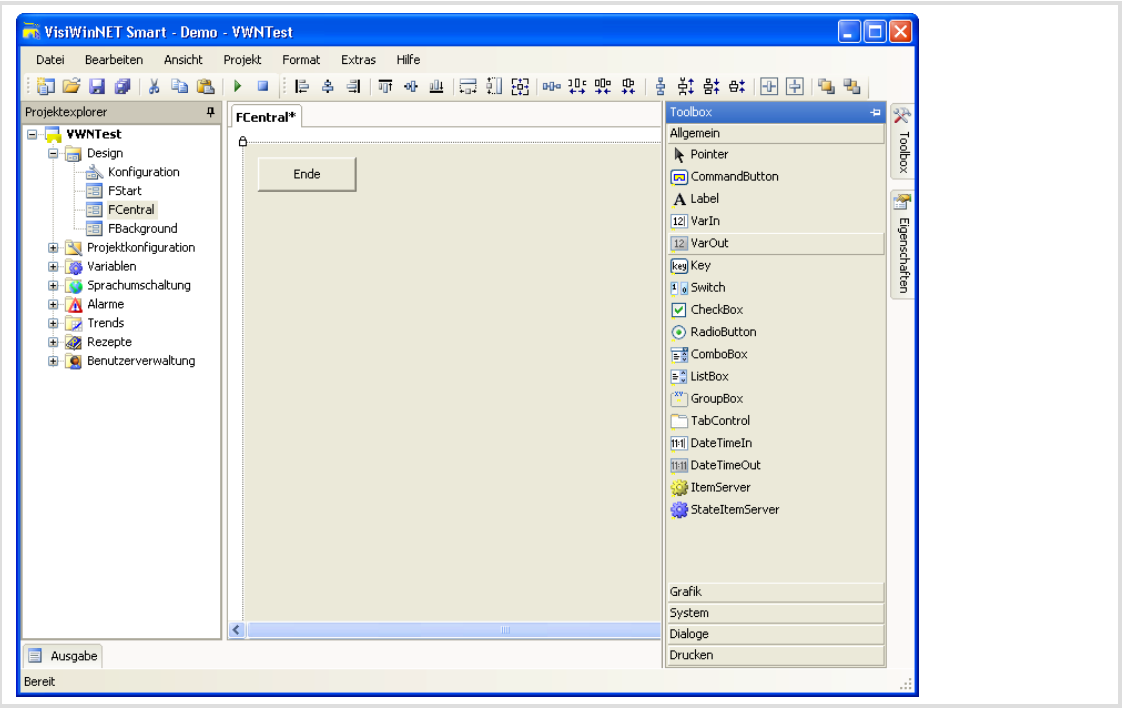
FCentral: Dieses Formular ist die zentrale Bildschirmansicht Ihres Laufzeitssystems. Von diesem Formular aus navigieren Sie zu weiteren Formularen und beenden die Applikation.

1. Öffnen Sie im Projektexplorer den Knoten **Design** und klicken Sie doppelt auf das Formular-Objekt **FCentral**.

   Das Formular stellt eine Bildschirmansicht dar, mit einem Steuerelemt des Typs ”Schaltfläche” (Ende) dar.

   Das Aussehen der Bildschirmansicht wird über das Platzieren und Parametrieren von Steuerelementen bestimmt. Steuerelemente werden in der Toolbox ausgewählt.

2. Öffnen Sie die Toolbox, indem Sie am oberen rechten Fensterrand auf den Karteireiter **Toolbox** klicken.
3. Markieren Sie unter **Allgemein** den Steuerelementtyp **VarOut**.



4. Ziehen Sie das Steuerelement bei gedrückter linken Maustaste von der Toolbox auf das Formular.

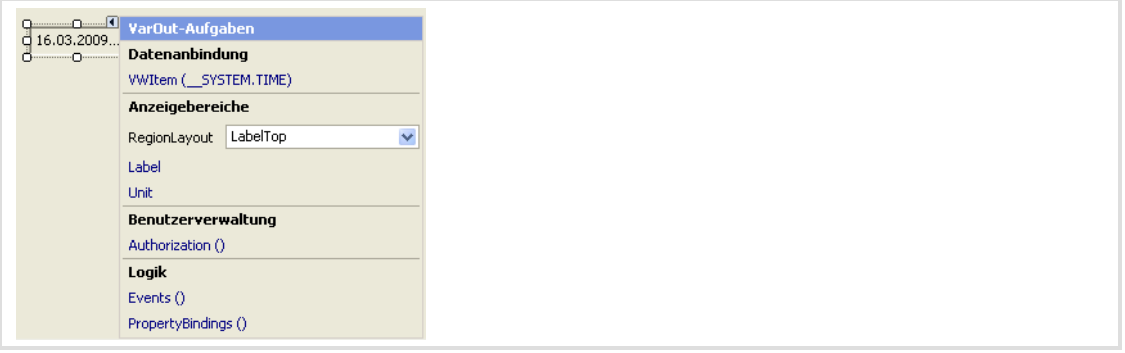
   Das Steuerelement wird mit einer vorgegebenen Größe platziert.

Tipp!

In der Standard-Einstellung wird das Toolbox-Fenster automatisch geschlossen, sobald Sie außerhalb der Toolbox klicken. Dieses Verhalten können Sie über einen Klick auf das Pinnwandnadel-Symbol ober rechts auf der Toolbar ändern:

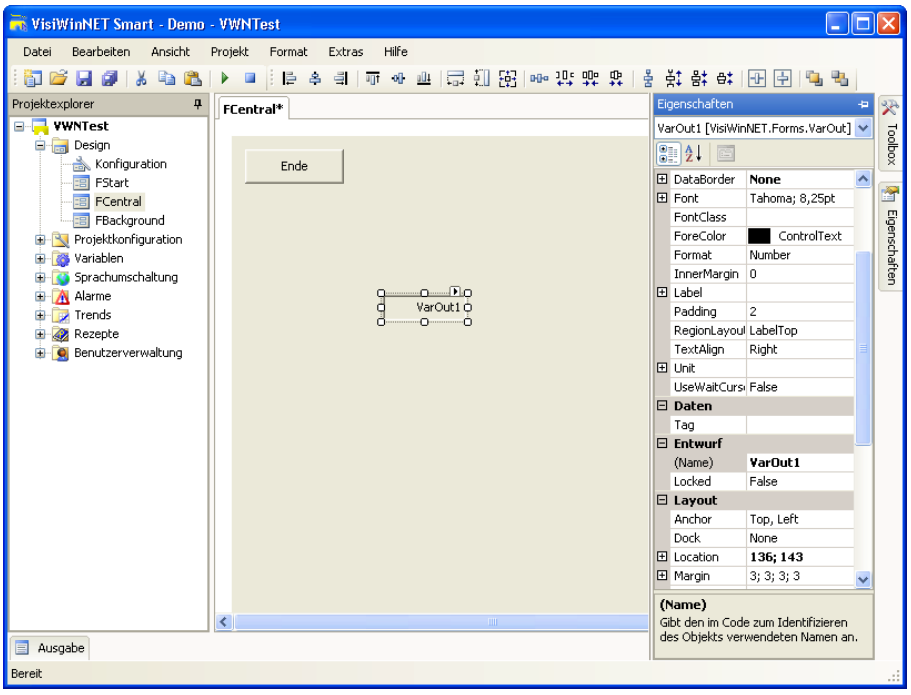
- ► Zeigt die Nadelspitze nach links, wird die Toolbox automatisch geschlossen.
- ► Zeigt die Nadelspitze nach unten, bleibt die Toolbox geöffnet.

Zeitgleich mit dem Platzieren, wird das zugehörige Aufgabenmenü geöffnet.



5. Schließen Sie das Aufgabenmenü, indem Sie außerhalb des Menüs mit der linken Maustaste klicken.

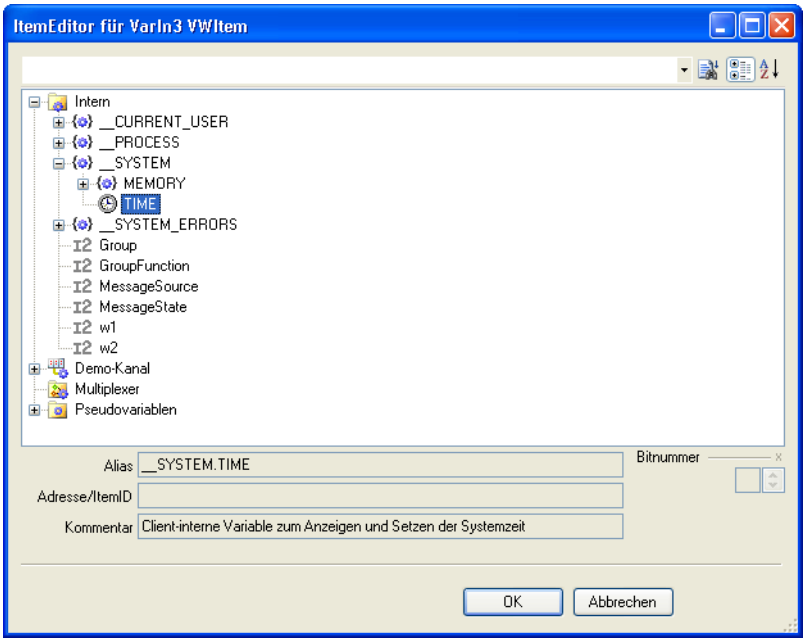
6. Markieren Sie das neue Steuerelement und öffnen Sie die zugehörigen Eigenschaftenliste, indem Sie am oberen rechten Fensterrand auf den Karteireiter **Eigenschaften** klicken.



Das Eigenschaftenfenster zeigt die aktuellen Parameter des Steuerelements.

Der gewählte Steuerelementetyp dient zur numerischen oder alphanumerischen Anzeige von Variablenwerten. Die wesentliche Festlegung, welcher Prozesswert anzuzeigen ist, wird über die Eigenschaft "VWItem" (am Ende der Liste) vorgenommen.

7. Markieren Sie im Eigenschaftenfenster-Abschnitt **VisiWinNET** die Eigenschaft **VWItem** und klicken Sie dort auf die Schaltfläche **[...]**.

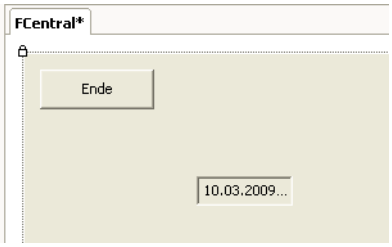


Der Dialog zeigt die zur Verfügung stehenden Prozessvariablen, die durch Browsen oder selbst Definieren in der Projektdatenbank gespeichert wurden. Im wesentlichen folgt der Aufbau der hier dargestellten Hierarchie dem Aufbau im Projektexplorer.

"Intern" kennzeichnet Variablen, die keinem Kommunikationskanal zugeordnet sind, die also kein reales Abbild in einer Steuerung haben. Diese Variablen können trotzdem innerhalb einer Visualisierung benötigt werden, um z. B. variablengesteuerte Vorgänge auszulösen oder einfach als Puffer für Werte aus der Visualisierung.

"Demo-Kanal" bzw. "Extern" beinhaltet die Variablen, die ein SPS-Abbild haben. Unter "Extern" werden die Kommunikationskanäle aufgelistet. Darunter kommen die Namensräume, die auch schon beim Browsen verwendet wurden.

8. Wählen Sie die Variable deren Wert Sie durch das Steuerelement anzeigen lassen möchten (z. B. "Intern" → "__System" → "Time") und klicken Sie auf **OK**.



Durch Festlegung der VWItem-Eigenschaft ändert sich die Darstellung im Steuerelement "VarOut1". Bei der Festlegung der darzustellenden Prozessvariablen wird der Datentyp interpretiert. Die Anzeige in den Steuerelementen springt auf einen Wert, der die Darstellung zur Laufzeit simuliert. Damit ist es möglich, schon zur Entwicklungszeit zu überprüfen, ob das Steuerelement den Wert überhaupt vollständig anzeigen kann oder ob ggf. die Steuerelementabmessung geändert werden muss.

9. Markieren Sie das Steuerlement **VarOut** und skalieren Sie es, bis auch die Uhrzeit dargestellt wird.
10. Speichern Sie das Formular über den Menüpunkt **Datei** → **Speichern** und schauen Sie sich das Ergebnis an, indem Sie das Projekt über **[F5]** starten.

Hinweis!

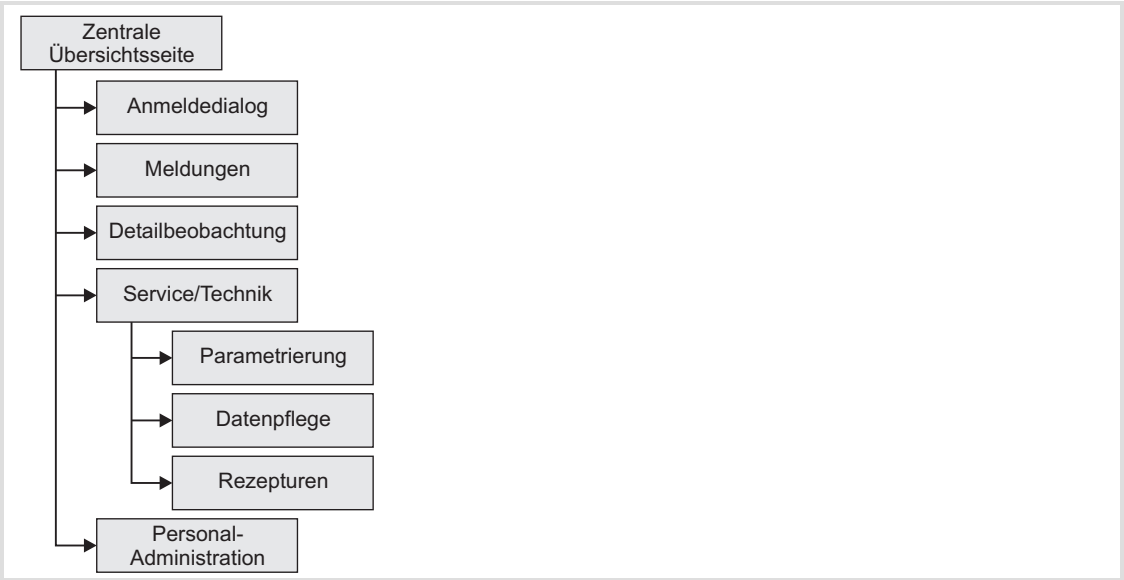
Wenn Sie im Demo-Modus (ohne Dongle) arbeiten, wird bei jedem Start des Laufzeitsystems ein Hinweis-Dialog eingeblendet. Schließen Sie ihn über die Schaltfläche **OK**.

4.4.2 Weitere Formulare erstellen und zwischen diesen navigieren

Selten wird eine Visualisierung mit nur einer Bildschirmansicht auskommen. Üblich und nützlich ist die Unterteilung in unterschiedliche Bildschirmansichten nach Funktion oder Logik der Anlage. Ausgehend von einer zentralen Übersichtseite kann der Benutzer dann über Schaltflächen in die ihn betreffenden Applikationsteile wechseln und dort seine Eingaben oder Beobachtungen vornehmen. Zu beachten ist dabei, die personenbezogene Freigabe von Funktionen. Allgemein üblich ist hier z. B. der Zugriff auf eine Service-Seite, die nicht vom Bedienpersonal, aber von einem Techniker aufgerufen werden darf.

Prinzipieller Aufbau einer Visualisierung

Eine wesentliche Überlegung beim Aufbau einer Visualisierung ist die Aufteilung der Bildschirmansichten. Ein typisches Beispiel soll hier gezeigt werden:



Jeder der hier dargestellten Funktionsblöcke stellt eine Bildschirmansichten dar. Die Hierarchie zeigt, wie die Formulare untereinander zu verknüpfen sind; das heißt, von wo welches Formular aufgerufen werden kann. Wenn z. B. aus der zentralen Übersicht auf die einzelnen darunter liegenden Funktionsblöcke zugegriffen werden soll, sind mindestens fünf Schaltflächen nötig.

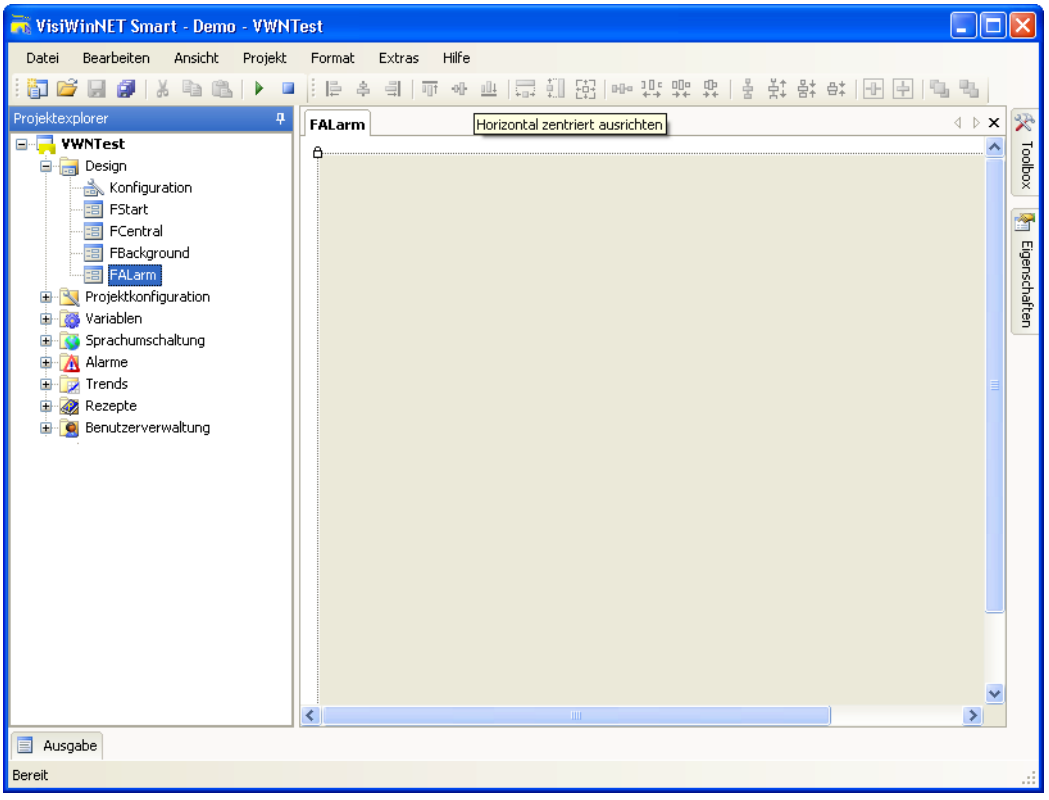
Geschmackssache ist, ob parallel zueinanderliegende Bildschirmseiten untereinander verknüpft werden müssen. Eine Ausnahme hierbei ist jedoch mit Sicherheit eine Alarmansichtenseite, die auf jeden Fall schnell aus allen Teilen der Applikation erreichbar sein sollte.

Neues Formular erstellen

1. Erstellen Sie ein neues Formular, indem Sie auf dem Knoten **Design** mit der rechten Maustaste klicken und im Kontextmenü dem Befehl **Formular hinzufügen** aktivieren.

   Ein neues Objekt mit dem Namen ”Form” wird unterhalb des Knotens ”Design” eingefügt und ein leeres Formular wird geöffnet.

2. Ändern Sie zunächst den Namen des Formulars indem Sie auf dem Formular-Objekt ”Form” mit der rechten Maustaste klicken, dort den Befehl **Umbenennen** wählen und den alten Text durch **FAlarm** ersetzen.



3. Öffnen Sie die Eigenschaften des Formulars indem Sie auf dem Formular-Objekt ”Form” mit der rechten Maustaste klicken und dort den Befehl **Eigenschaften** wählen.

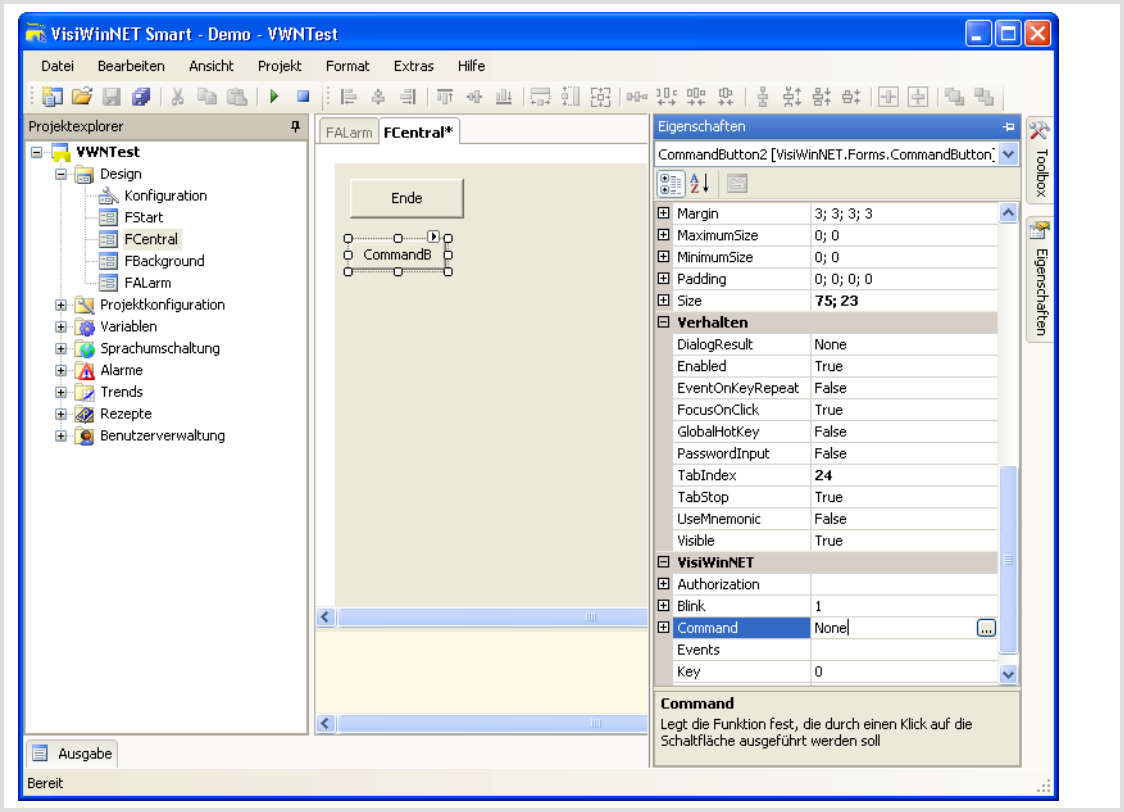
   Das Eingenschaften-Fenster wird geöffnet. Hier können Sie folgende Eingenschaften konfigurieren:

   – ExecuteCode
   – LoadOnStartup
   – VariableActivationMode
   – VariableDeactivationMode

Schaltfläche zum Navigieren zwischen Formularen erstellen

1. Wechseln Sie über die Karteireiter auf das Formular-Objekt **FCentral**.
2. Klicken Sie oben rechts auf den Karteireiter **Toolbox**, um die Toolbox anzuzeigen.
3. Markieren Sie unter der Toolbox-Sektion **Allgemein** den Steuerelementtyp **CommandButton**.
4. Klicken Sie im Formular an die Stelle, wo das Steuerelement eingefügt werden soll.
5. Markieren Sie das neue Steuerelement und klicken Sie oben rechts auf den Karteireiter **Eigenschaften**, um das Eigenschaftenfenster anzuzeigen.

   Das Eigenschaftenfenster zeigt die aktuellen Parameter des Steuerelements.

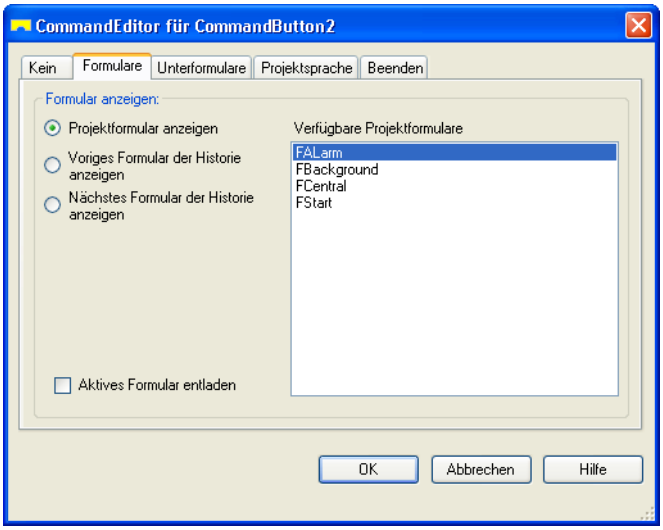


   Der gewählte Steuerelementetyp dient zur Ausführung eines Befehls. Dieser wird über die Eigenschaft ”Command” definiert.

6. Markieren Sie in der Eigenschaften-Sektion **VisiWinNET** die Eigenschaft **Command** und klicken Sie dort auf die Schaltfläche **[...]**.

   Der Dialog ”Command’Editor ...” zeigt die zur Verfügung stehenden Befehle, geordnet in Register.

7. Aktivieren Sie das Register **Formulare**.



8. Markieren Sie das Optionsfeld **Projektformulare anzeigen** und wählen Sie im Auswahlfeld **Verfügbare Projektformulare** das Formular-Objekt **FAlarm**. Schließen Sie den Dialog über **OK**.
9. Ändern Sie jetzt noch die Beschriftung der Schaltfläche, indem Sie in der Eigenschaft **Text** die Beschriftung **FAlarm** eintragen.

   Bevor Sie das Projekt testen: Sie haben jetzt eine Schaltfläche definiert, über die Sie während der Laufzeit vom Formular ”FCentral” zum Formular ”FAlarm” springen können. Was aber noch fehlt, ist eine Möglichkeit wieder zurück zu springen.

Hinweis!

Wenn Sie keine Möglichkeit schaffen, zurück zu springen, können Sie das Laufzeitsystem nicht beenden. Es bleibt dann nur, VisiWinNET®Smart über den Windows-Taskmanager zu beenden.

Sie könnten jetzt, analog zur beschriebenden Vorgehensweise, auf dem Formular ”FAlarm” eine Schaltfläche positionieren, die dann wieder zum Formular ”FCentral” zurückspringt. Wir wollen Ihnen aber dafür eine Alternative aufzeigen.

10. Wechseln Sie zum Formular **FAlarm**, markieren Sie das Formular und öffnen Sie das Eigenschaften-Fenster.
11. Wählen Sie unter **FormBorderStyle** den Eintrag **FixedDialog**.

    Sie haben damit dem Formular das Aussehen und die Funktion eines Standard-Windows-Fenster zugewiesen, oben rechts mit den drei bekannten Schaltflächen zum Minimieren, Skalieren und Schließen von Fenstern.
12. Speichern Sie das Formular über den Menüpunkt **Datei** → **Speichern** und schauen Sie sich das Ergebnis an, indem Sie das Projekt über **[F5]** starten.

4.5 Benutzer verwalten

Die VisiWinNET® Smart-Benutzerverwaltung dient zur Vergabe von personenbezogenen Berechtigungen innerhalb der Visualisierungsapplikation. Eingabesteuerelemente können gesperrt, der Zugang zu Informationen über Ausgabesteuerelemente kann verweigert werden. VisiWinNET® Smart unterstützt zwei Verwaltungssysteme für Benutzer:

Ebenenorientierte Benutzerverwaltung: Benutzer- und Steuerelemente können einer von 999 Ebenen zugeordnet werden. Ist zur Laufzeit die Ebene eines Steuerelementes höher als die des gerade angemeldeten Benutzers, so wird die Bedienung des Steuerelementes abgelehnt.

Rechteorientierte Benutzerverwaltung: Den Steuerelementen in der Applikation wird ein Recht zugeordnet, das der Benutzer besitzen muss, um das Steuerelement bedienen zu können (z. B. ”Rezept bearbeiten”). Anschließend werden den Benutzern die Rechte zugeteilt, die sie ausführen dürfen. Dies ist insbesondere in einem Client-Server-System sinnvoll, wenn mehrere Mitarbeiterhierarchien (Gruppen von Mitarbeitern, die unabhängig voneinander in Berechtigungsebenen aufgeteilt sind) im Netzwerk arbeiten. Über den Vergleich der Rechte von Benutzern und Steuerelementen wird über die Freigabe entschieden.

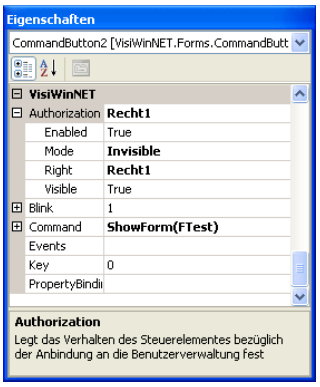
Die Verwaltung von Benutzern wird über drei Definitionen in der Projektdatenbank vorgenommen:

Benutzergruppe: Fasst Benutzer zu einer Gruppe zusammen und enthält die charakteristischen Parameter für alle enthaltenen Benutzer. Über die Benutzergruppe werden die Bezüge auf die zur Bedienung der Applikation nötigen Rechte verwaltet.

Benutzer: Enthält die Zugangsinformationen (Anmeldename und Passwort) und den Aktivierungsstatus eines Benutzers.

Recht: Rechte definieren die Information, die den Steuerelementen der Applikation zugeordnet werden. Über die ”Authorization”-Eigenschaft wird einem Steuerelement ein Recht zugeordnet. Ein angemeldeter Benutzer muss in seiner Benutzergruppendefinition einen Verweis auf dieses Recht haben, damit er das Steurelement bedienen darf. Rechte werden nur in der rechteorientierten Benutzerverwaltung verwendet. In der ebenenorientierten Benutzerverwaltung erwartet die ”Authorization”-Eigenschaft der Steuerelemente eine Zahl.

Alle VisiWinNET® Smart-Steuerelemente beinhalten eine ”Authorization”-Eigenschaft. In dieser Eigenschaft wird der Name des Rechtes festgelegt, durch das ein Steuerelement zur Laufzeit bedienbar sein soll. Außerdem wird spezifiziert, wie sich das Steuerelement bei einer fehlenden Berechtigung verhält. Wird die Eigenschaft im Eigenschaftenfenster expandiert, kann in der darunterliegenden ”Mode”-Einstellung zwischen ”disabled” (deaktiviert) und ”invisible” (unsichtbar) gewählt werden.



Zur Laufzeit stellt VisiWinNET® Smart spezielle Dialoge zur Verwaltung der Benutzerinformationen zur Verfügung. Ein typisches Beispiel ist der Anmeldedialog, der als Eingaben den Benutzernamen und ein Passwort verlangt. Über die hier vorgenommene Anmeldung werden die Steuerelemente automatisch, je nach vergebener Berechtigung, aktiviert oder deaktiviert. Der Aufruf der Benutzerverwaltungsdialoge wird durch die Komponente ”UserManagementDialogs” unterstützt.

Das folgende Beispiel zeigt die Arbeitsschritte zur Implementierung einer rechteorientierten Benutzerverwaltung. Beispielhaft wird gezeigt, wie die Verknüpfung der einzelnen Bestandteile auszuführen ist und wie zur Laufzeit Benutzerdaten angepasst werden können. Enthalten sind ein Administrator sowie zwei Benutzer mit unterschiedlichen Rechten.

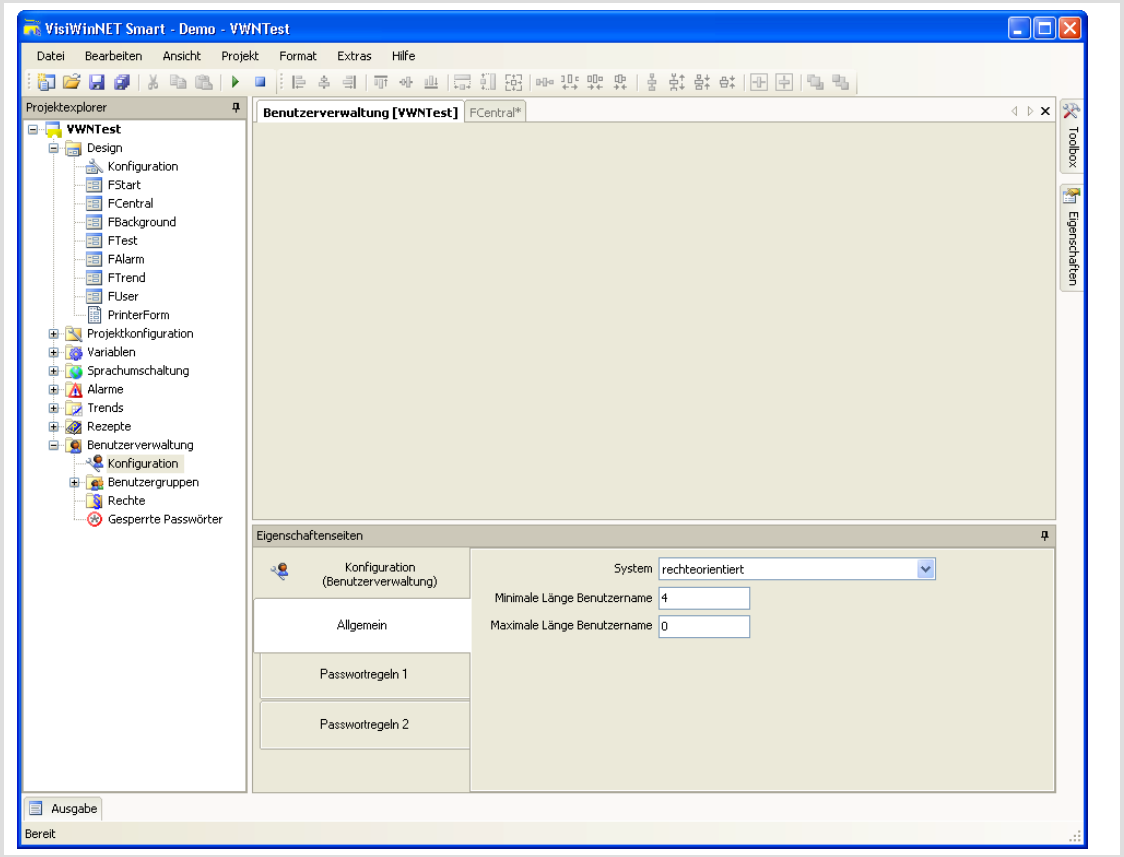
Mithilfe der ”Authorization”-Eigenschaft werden die entsprechenden Freigaben in den Steuerelementen verknüpft.

Außerdem beinhaltet das Beispiel den Aufruf des Anmeldedialoges und des Administrationsdialoges zur Pflege von Benutzerdaten.

4.5.1 Benutzerverwaltungssystem festlegen

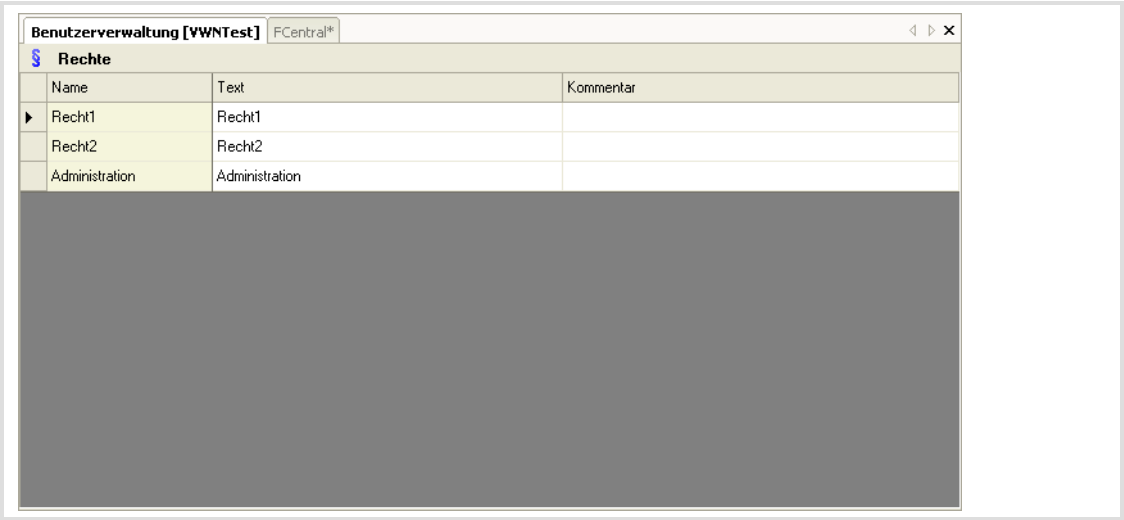
1. Markieren Sie im Projektexplorer den Knoten **Benutzerverwaltung** → **Konfiguration** und geben Sie auf der Eigenschaftenseite unter **System** den Wert **rechteorientiert** vor.

   Im Projektexplorer wird der Knoten ”Rechte” angezeigt.



4.5.2 Rechte anlegen

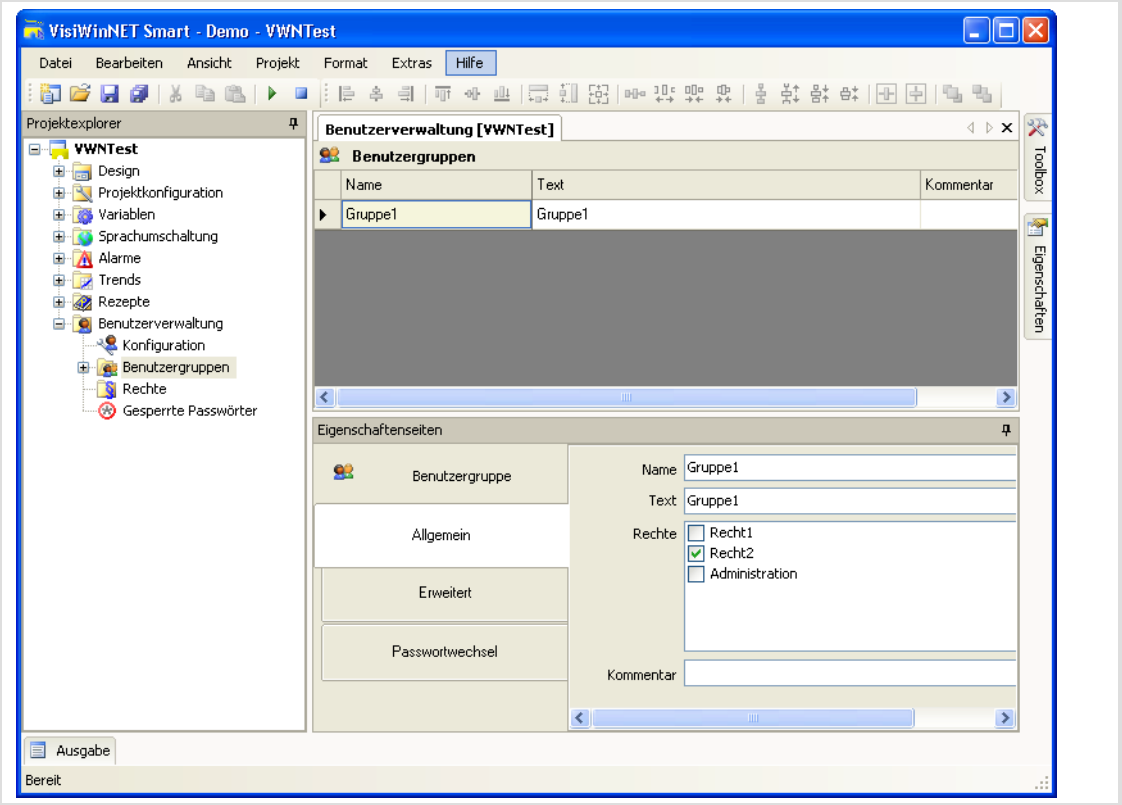
1. Klicken Sie doppelt auf den den Knoten **Rechte**.
2. Klicken Sie im Clientbereich mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.
   Im Tabelleneditor wird ein neues Recht erzeugt (”Recht1”).
3. Erzeugen Sie auf gleiche Weise zwei weitere Rechte (”Recht2” und ”Administration”).



4.5.3 Benutzergruppen anlegen

Die Zuordnung Benutzer / Rechte erfolgt über die Benutzergruppe. Das erscheint zwar auf den ersten Blick etwas kompliziert, hilft aber in der realen Welt: Jede Benutzergruppe kann beliebig viele Benutzer aufnehmen. Für alle enthaltenen Benutzer gemeinsam sind die Rechte in der Gruppe einstellbar, also gemeinsam anzupassen.

1. Markieren Sie im Projektexplorer den Knoten **Benutzergruppen**.
2. Erzeugen Sie eine Benutzergruppe. Klicken Sie dazu im Clientbereich mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.



3. Erzeugen Sie zwei weitere Benutzergruppen.

4. Weisen Sie den drei Benutzergruppen folgende Eigenschaften zu:

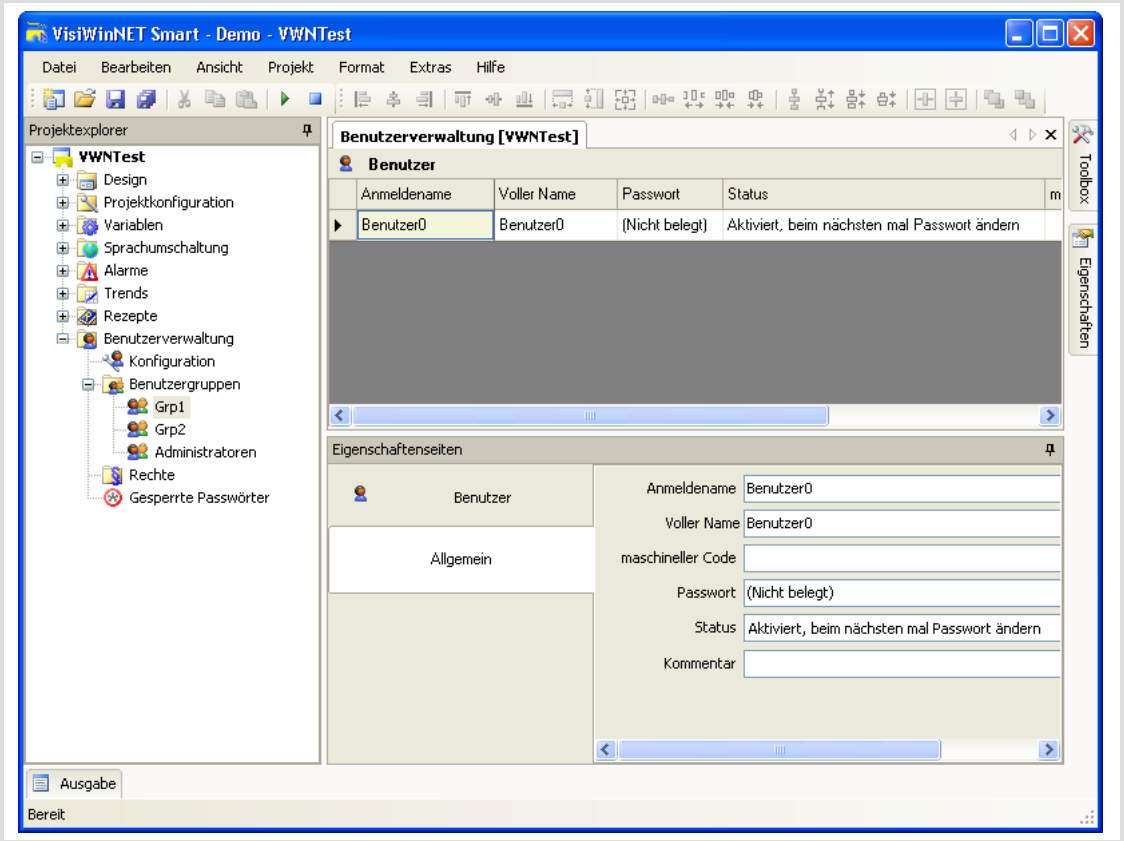
Eigenschaften für Benutzergruppen		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Name	Grp1
	Text	Grp1
	Recht	Recht1
2	Allgemein	
	Name	Grp2
	Text	Grp2
	Recht	Recht2
3	Allgemein	
	Name	Administratoren
	Text	Administratoren
	Recht	Recht1, Recht2, Administration

Administratoren dürfen etwas mehr, in diesem Fall alles.

4.5.4 Benutzer anlegen

1. Öffnen Sie im Projektexplorer den Knoten **Benutzergruppen** und klicken Sie auf das Element **Grp1**.
2. Klicken Sie im Clientbereich mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.

   Im Tabelleneditor wird ein neuer Benutzer erzeugt (”Benutzer0”). Dieser Benutzer ist Mitglied der Benutergruppe ”Grp1”.



3. Legen Sie in der Benutzergruppe ”Grp1” und ”Administratoren” auch einen Benutzer an.

4. Weisen Sie den Benutzern auf der jeweiligen Eigenschaftenseite die folgenden Eigenschaften zu:

Eigenschaften für Benutzer		
Gruppe	**Register/Eigenschaft**	**Wert**
Grp1	Allgemein	
	Name	Max
	Vollständiger Name	Max Mustermann
	Passwort	0000
	Status	aktiviert
Grp2	Allgemein	
	Name	Moritz
	Vollständiger Name	Moritz Mustermann
	Passwort	0000
	Status	aktiviert
Admin.	Allgemein	
	Name	Admin
	Vollständiger Name	Admin Mustermann
	Passwort	0000
	Status	aktiviert

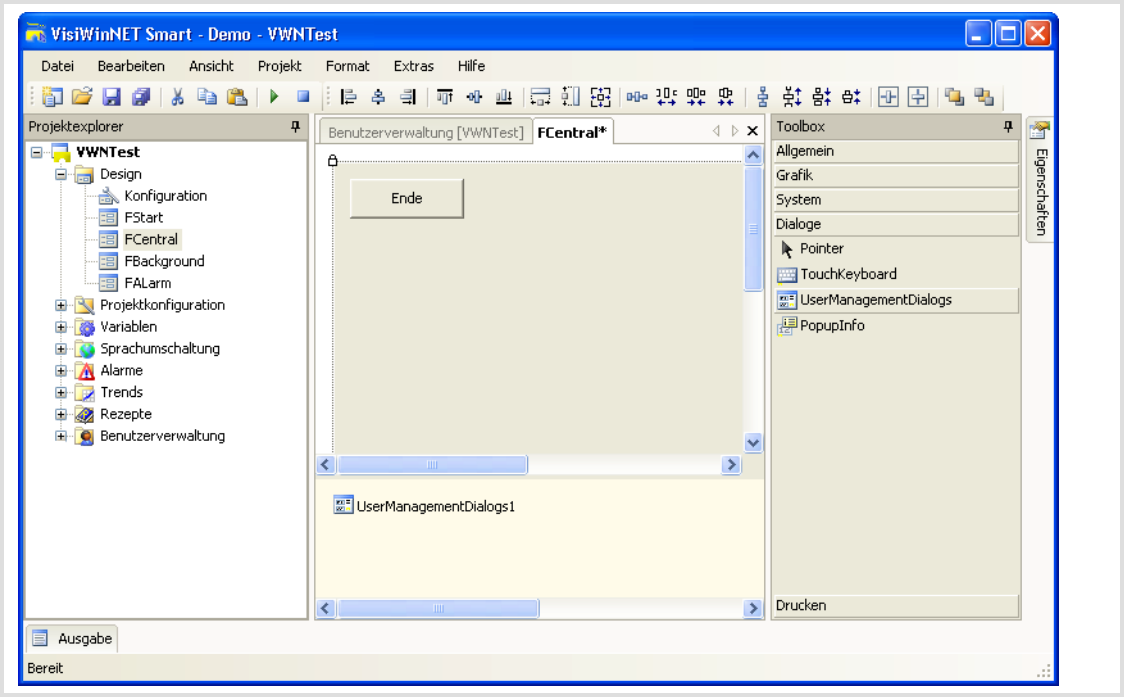
Anmeldename und Passwort werden später als Anmeldeinformationen verwendet. Der Status erlaubt verschiedene Einstellungen, die darüber entscheiden, ob der Benutzer sich anmelden darf und ob er das vom Projekteur vergebene Passwort beim ersten Anmelden ändern muss.

4.5.5 Dialoge für die Benutzerverwaltung einsetzen

1. Öffnen Sie das Formular **FCentral** und platzieren Sie dort ein Toolbox-Steuerelement vom Typ ”UserManagementDialogs” (Abschnitt ”Dialogs”).

   Damit stellen Sie dem Projekt diverse Standard-Dialoge für die Benutzerverwaltung zur Verfügung.

   Nachdem das Steuerelement auf dem Formular platziert wurde, rutscht es automatisch in das sogenannte ”Komponentenfach” unterhalb des Formulars.



2. Platzieren Sie auf dem Formular **FCentral** vier Steuerelemente vom Typ ”CommandButton” mit folgenden Eigenschaften:

Eigenschaften für Steuerelemente			
Nr.	**Steuerelementtyp/Eigenschaft**	**Wert**	**Bemerkung**
1	CommandButton		
	Darstellung/Text	LogOn	
	VisiWinNET/Authorization		leer = für alle Benutzer zugänglich
	VisiWinNET/Events	Click-Ereignis mit dem Aufruf der **ShowLogOnDialog**-Funktion der **UserManagementDialogs1**-Komponente verknüpfen.	Über die Schaltfläche **[...]** öffnet sich ein Dialog. Hier ist das Click-Ereignis auszuwählen und die Schaltfläche **Auswählen** zu drücken.
2	CommandButton		
	Darstellung/Text	Grp1	
	VisiWinNET/Authorization	Recht1	Nur für Benutzer der Benutzergruppe mit Recht 1 (=Grp1) zugänglich
3	CommandButton		
	Darstellung/Text	Grp2	
	VisiWinNET/Authorization	Recht2	
4	CommandButton		
	Darstellung/Text	Admin	
	VisiWinNET/Authorization	Administration	
4	VisiWinNET/Events	Click-Ereignis mit dem Aufruf der **ShowUsersDialog**-Funktion der **UserManagementDialogs1**-Komponente verknüpfen.	

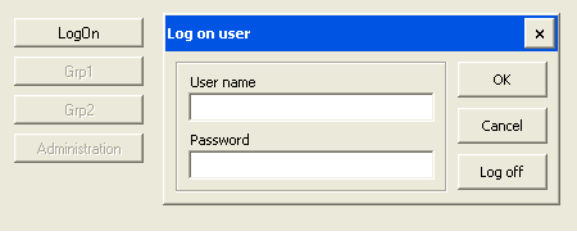
Die Text-Eigenschaft dient hier zur Anzeige der Schaltflächenfunktion. Die ”Authorization”-Eigenschaft legt jeweils fest, wann die Schaltfläche aktiviert sein soll: Ist ein Benutzer angemeldet und ist seine Benutzergruppe mit dem hier angegebenen Recht verknüpft, wird die Schaltfläche aktiviert.

Die Einstellung der ”Events”-Eigenschaft dient zum Aufruf der Dialoge ”Benutzer anmelden” und ”Benutzer administrieren”.

3. Starten Sie das Projekt mit **[F5]**.

   Die Schaltfläche ”LogOn” ist freigeschaltet. Hier wurde keine Einstellung in der ”Authorization”-Eigenschaft vorgenommen.

4. Klicken Sie auf die Schaltfläche **LogOn**.



Der Dialog zum Anmelden eines Benutzers wird geöffnet. Die drei Schaltflächen ”Grp1”, ”Grp2” und ”Administration” werden durch die Anmeldung des entsprechenden Benutzers freigeschaltet.

5. Probieren Sie Ihre Benutzernamen mit ihren Passwörtern aus.

   Die beiden Schaltflächen ”Grp1” und ”Grp2” sind funktionslos. Sie werden jeweils über die Benutzer aus den Gruppen ”Grp1” und ”Administration” bzw. ”Grp2” und ”Administration” freigeschaltet. Werden so parametrierte Schaltflächen dazu verwendet, zu weiteren Formularen zu wechseln, können ganze Applikationsteile in Abhängigkeit vom aktuellen Benutzer freigegeben werden.

   Die Schaltfläche ”Administration” steht nur Benutzern aus der gleichnamigen Benutzergruppe zur Verfügung. Ein Klick auf diese Schaltfläche öffnet den Benutzerverwaltungsdialog. Hier können neue Benutzer angelegt und bestehende verwaltet werden.

4.6 Mehrsprachige Projekte erstellen (Sprachumschaltung)

Die Sprachumschaltung ermöglicht den Wechsel sämtlicher in der Applikation verwendeter Texte in Abhängigkeit von der gewählten Sprache. Sie wird zum Beispiel verwendet, um ...

- eine internationale Applikation auszuliefern. Hier würde genügen, die anzuzeigende Sprache beim Projektstart oder auf einer Serviceseite zu konfigurieren.
- internationalem Bedienpersonal die Bedienung zu ermöglichen. Denkbar ist hier eine Funktion, dass beispielsweise beim Schichtwechsel die Sprache umgestellt werden kann, wenn sich die Nationalität des Personals ändert.

Beides ist auf Knopfdruck möglich.

Neben den eigentlichen Texten, die in der Applikation verwendet werden, sind weitere Besonderheiten zu beachten:

- In Abhängigkeit von der gewählten Sprache müssen
  – Einheiten (z. B. Temperatur °C/°F) umgerechnet und umgestellt werden
  – Datums- und Zeitformate im landesüblichen Format dargestellt werden.
- Gegebenenfalls ist die Größe von Schriftarten in unterschiedlichen Zeichensätzen anders normiert, so dass beim Wechsel der Sprache eine Anpassung der Schriftart nötig wird.

Weitere Informationen zum Thema ”Entwicklung mehrsprachiger Visualisierungsapplikationen” finden Sie im Anhang (📖 127).

4.6.1 Sprachumschaltung aufrufen

1. Expandieren Sie im Projektexplorer den Knoten **Sprachumschaltung**.

   Folgende Unterknoten werden angezeigt:

   Der **Konfiguration**-Knoten beinhaltet folgende Untergruppen:

   – **Fontklassen** und **Fonts** definieren die Funktion der Schriftartenumschaltung.

   – **Sprachen** beinhaltet als Liste die Projektsprachen. Ein neues Projekt beinhaltet die Sprachen Deutsch und Englisch. Weitere Sprachen können dem Projekt hinzugefügt werden

   Benutzertexte sind die Texte, die der Anwender anlegen kann, um sie in der Applikation anzuzeigen.

   Komponenten und **Dialoge** enthalten die Texte, die von den Steuerelementen und Dialogen des VisiWinNET-Pakets verwendet und erwartet werden. Hier können keine Texte angelegt werden, die Texte können sprachlich erweitert oder applikationsbezogen geändert werden.

   In den Unterknoten **Variablen**, **Alarme**, **Archive**, **Benutzerverwaltung** und **Protokollierung** können keine Texte angelegt werden. Das wird von den Editoren der entsprechenden Systeme erledigt. Wird beispielsweise ein Alarm im Alarmsystem angelegt, erscheint der sprachumschaltbare Text des Alarms unter dem hier angezeigten Knoten. Dies entspricht dem zentralen Verwaltungsgedanken der Sprachumschaltung

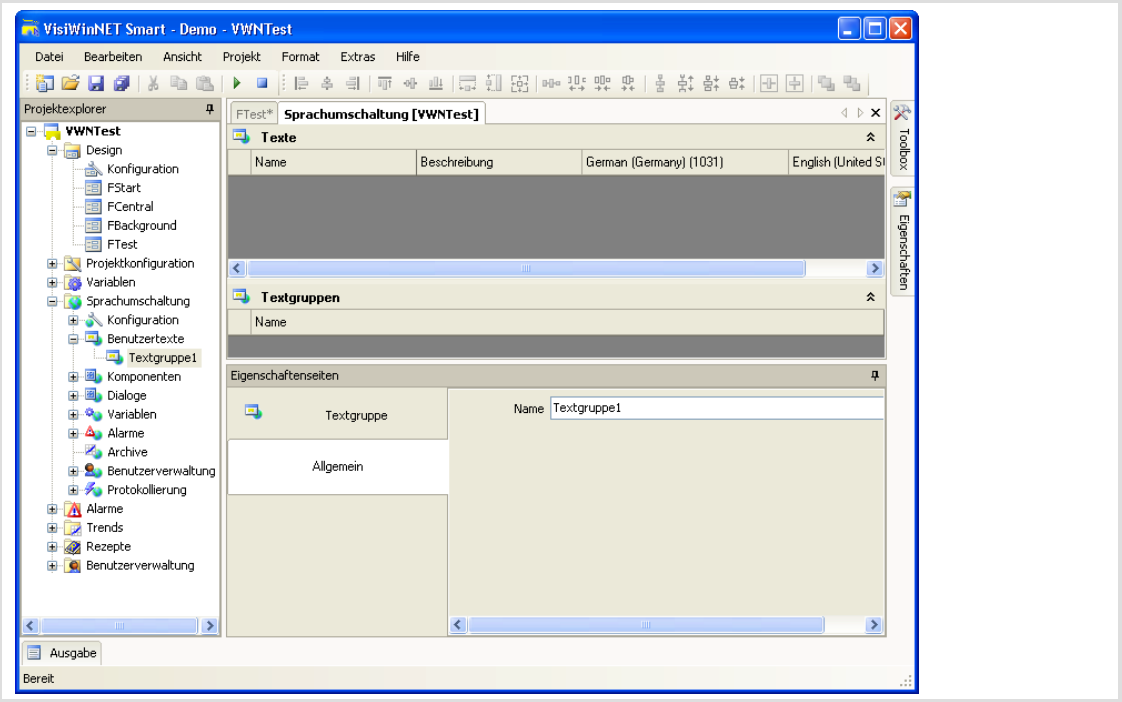
Hinweis!

Die Einheitenumschaltung ist in der Prozessanbindung enthalten. Sie ist konzeptionell von der Sprachumschaltung getrennt worden, da die zu definierenden Informationen als Hauptmerkmal Umrechnungswerte enthalten.

4.6.2 Benutzertexte anlegen

1. Klicken Sie im Projektexplorer auf den Knoten **Sprachumschaltung** → **Benutzertexte** mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.

   Unterhalb des Knotens Benutzertexte wird der Unterknoten ”Textgruppe1” erstellt. Im Clientbereich wird die Objektliste ”Texte” und ”Textgruppe” angezeigt.



2. Ändern Sie zunächst auf der Eigenschaftenseite den Namen der Textgruppe in **FCentral**.

   Sprachumschaltbare Texte können in einzelnen Gruppen angelegt und verwaltet werden. Dies hat folgende Vorteile:

   – Über den Namen der Textgruppe kann angegeben werden, wo die Texte verwendet werden. Über Textgruppen kann also eine Gliederung erfolgen.
   – Die Texte, die angelegt und später vielleicht einmal geändert werden sollen, können leichter wiedergefunden werden.
   – Wird zu jedem Formular eine Textgruppe angelegt, erleichtert dies später die Arbeit. Wenn das Formular in ein anderes Projekt übernommen wird, muss nur die entsprechende Textgruppe in das Projekt kopiert werden. Dann ist das Formular, das die Texte verwendet, wieder funktionsfähig.

3. Klicken Sie auf der Objektliste **Text** mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.

   In der Objektliste ”Text” wird eine neue Zeile angelegt.

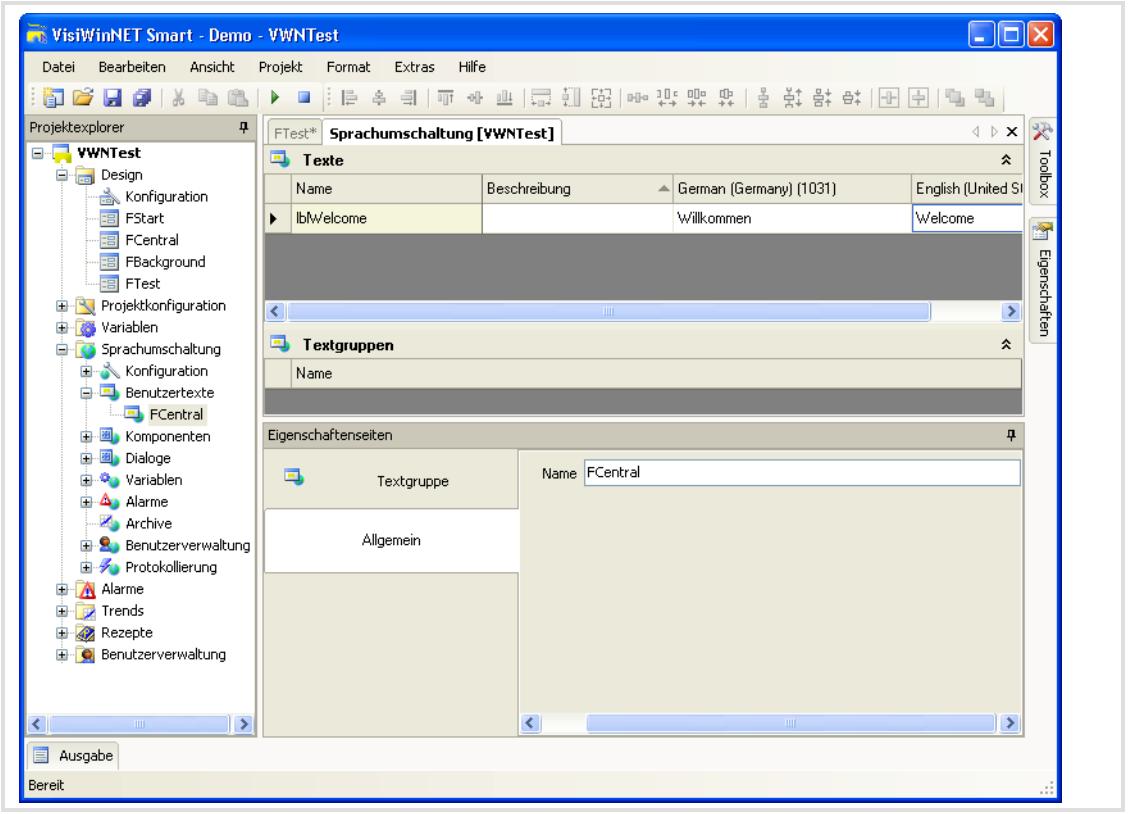
4. Geben Sie in dieser Zeile folgende Texte ein:

   Name = lblWelcome

   Beschreibung = *beliebiger Text oder leer*

   German = Willkommen

   Englisch = Welcome



Sie haben damit Ihren ersten Benutzertext definiert.

4.6.3 Benutzertexte einem Steuerelement vom Typ ”Label” zuweisen

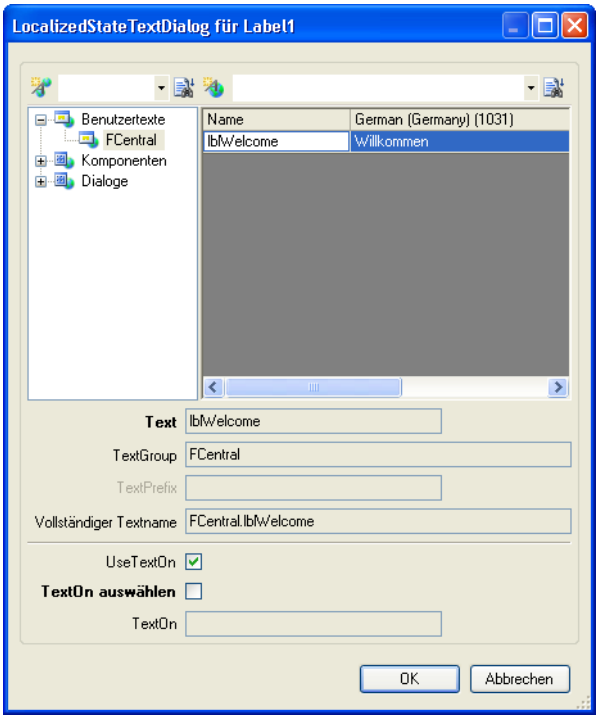
Als Nächstes werden wir auf dem Formular ”FCentral” ein Steuerelement vom Typ ”Label” erzeugen und es mit dem Benutzertext verknüpfen.

1. Expandieren Sie im Projektexplorer den Knoten **Design** und klicken Sie doppelt auf das Formular-Objekt **FCentral**.
2. Ziehen Sie das Steuerelement **Label** von der Toolbox auf das Formular **FCentral**.
   Das Steuerelement wird auf dem Formular angezeigt und das Aufgabenmenü ”Label-Aufgabe” wird geöffnet.

Hinweis!

Das Aufgabenmenü wird geschlossen, sobald Sie außerhalb des Menüs mit der linken Maustaste klicken. Sie können es bei markiertem Steuerelement über die kleine Schaltfläche ▶ am oberen Steuerelement-Rand erneut öffnen.

3. Klicken Sie im Aufgabenmenü **Label-Aufgabe** auf **LocalizedText (Label1)**.
   Der Dialog ”LocalizeStateTextDialog für Label1” wird geöffnet.
4. Öffnen Sie im Auswahlfeld **Textgruppe** den Knoten **Benutzertexte** und klicken Sie doppelt auf das Textgruppen Objekt **FCentral**, welches wir zuvor angelegt hatten.
   Im Listenfeld wird der Benutzertext ”lblWelcome” angezeigt.



5. Markieren Sie im Listenfeld den Text **lblWelcome** und klicken Sie auf **OK**.

   Sie haben damit festgelegt, welcher Text vom Steuerelement zur Laufzeit angezeigt werden soll. Nach der Übernahmen, wird im Steuerelement der Text in der aktiven Sprache angezeigt.

Hinweis!

Neben dem Label-Steuerelement unterstützen auch andere Steuerelemente direkt oder indirekt die Anzeige von sprachumschaltbaren Texten. Generell gilt: Eine ”LocalizedText”-Eigenschaft kennzeichnet die Anzeige eines sprachumschaltbaren Textes.

Andere Steuerelemente haben keine solche Eigenschaft, beziehen aber Texte oder Formatierungsanweisungen aus der Sprachumschaltung. Das ”AlarmLine”-Steuerelement beispielsweise beinhaltet die Eigenschaft ”DateTimeFormat”’, die die Angabe eines Textes aus der Sprachumschaltung erlaubt. Die Angabe ”@LongDate” als Eigenschaftenwert besagt, dass der gleichnamige Text aus der Textgruppe ”Komponenten.Time.DateFormats” für jede Projektsprache eine Formatierungszeichenkette enthält, die zur Laufzeit dazu verwendet wird, Datums-/Zeitwerte zu formatieren. Auch dies ist eine Funktion der Sprachumschaltung.

4.6.4 Sprache während der Laufzeit umschalten

Jetzt wollen wir dem Formular ”FCentral” zwei Schaltflächen zuweisen, über die die Sprache zur Laufzeit umgeschaltet werden kann.

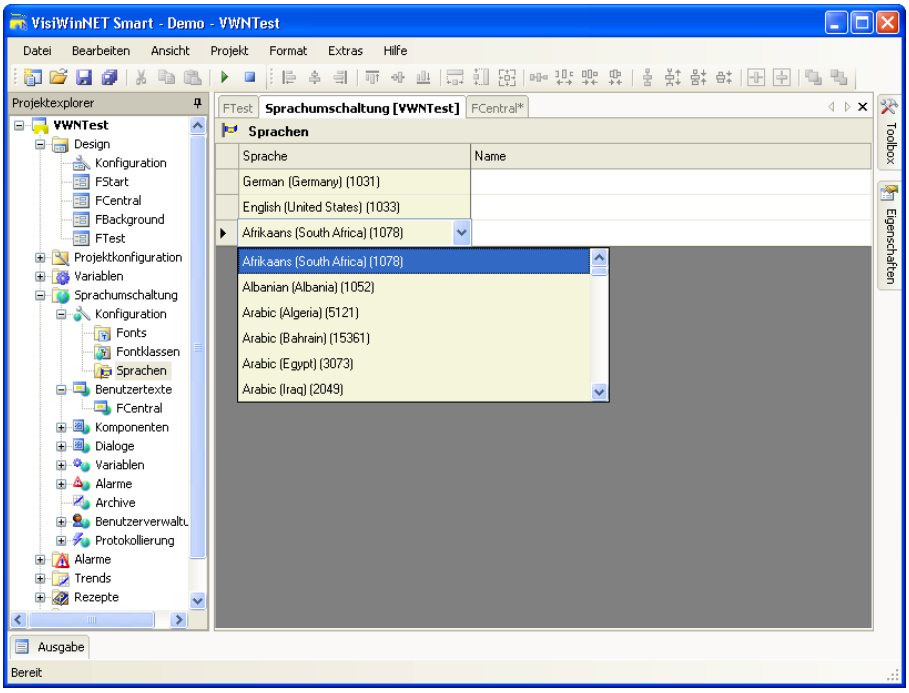
1. Ziehen Sie das Steuerelement **CommandButton** von der Toolbox auf das Formular **”FCentral”**.
   Das Steuerelement wird auf dem Formular angezeigt und das Menü ”CommandButton Task” wird geöffnet.
2. Schließen Sie das Menü **CommandButton Task** indem Sie mit der Maus auf das Formular klicken.
3. Markieren Sie das neue Steuerelement und öffnen Sie das Eigenschaftenfenster.
4. Ändern Sie im Eigenschaftenfenster den Namen des Steuerelements:
   Darstellung → **Text:** Englisch
5. Klicken Sie im Eigenschaftenfenter unter **VisiWinNET** → **Command** auf **[...]**.
   Der Dialog ”CommandEditor for ...” wird geöffnet.
6. Aktivieren Sie das Register **Projektsprache** und wählen Sie dort unter **Verfügbare Projektsprachen** den Eintrag **Englisch (United States)**.
7. Schließen Sie den Dialog über **OK**.
   Im Eigenschaftenfester unter **VisiWinNET** → **Command** steht jetzt der Eintrag ”ChangeLanguage(1033)”. Sie haben damit dieser Schaltfläche das Kommando zur Umschaltung der Benutzertexte auf die Sprache ”Englisch” zugewiesen. (1033 ist der ”Locale Identifier” für die Sprache amerikanisches Englisch; 1031 steht für Deutsch.)
8. Wiederholen Sie die Schritte, um eine weitere Schaltfläche anzulegen, über die Sie auf die Benutzersprache Deutsch zurückschalten können.

9. Drücken Sie **[F5]**, um das Projekt zu starten und die Funktionen zu testen.

4.6.5 Weitere Sprachen definieren

1. Um weitere Sprachen verfügbar zu machen, klicken Sie im Projektexplorer doppelt auf den Knoten **Sprachumschaltung** → **Konfiguration** → **Sprache**.
2. Klicken Sie im Clientbereich mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.

   Eine neue Zeile für eine neue Sprache wird erzeugt.
3. Öffnen Sie in der neuen Zeile das Auswahlfeld und wählen Sie die gewünschte Sprache.



Hinweis!

Das genannte Eingabegebietsschema ist nicht vorhanden?

Eingabegebietsschemata sind Unterstützungen des Betriebssystems, die die länderspezifischen Schrift-/Sonderzeichen auf der Tastatur umschalten.

So beinhaltet z. B. eine deutsche Tastatur die Umlaute ä, ö, und ü, die im englischen nicht existieren. Auf englischen Tastaturen sind diese Tasten mit anderen Zeichen belegt. Der gesendete Tastencode der Tasten ist jedoch identisch.

Das Betriebssystem regelt die Zuordnung der Tasten zu den auf der Tastatur aufgedruckten Zeichen über eine länderspezifische Tabelle, dem ”Eingabegebietsschema”.

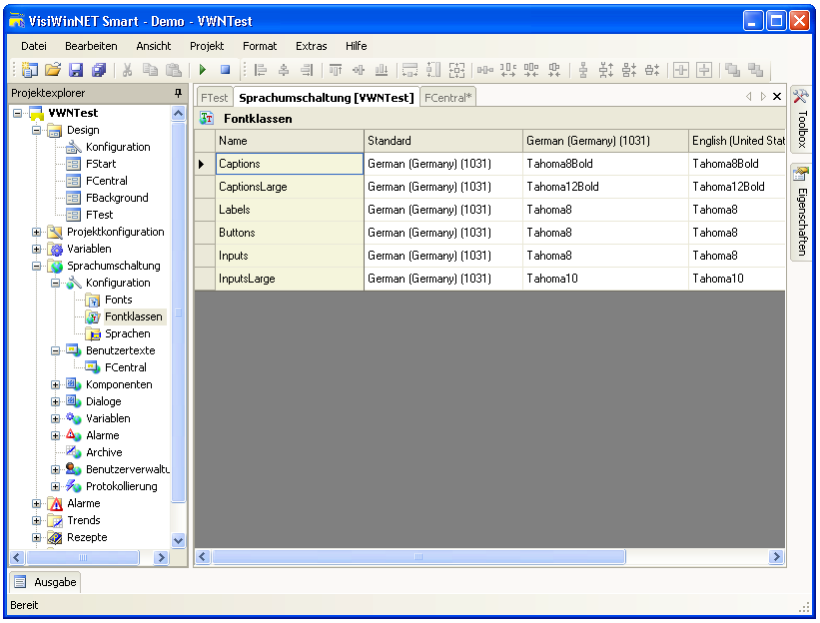
Benötigte Eingabegebietsschemata müssen installiert werden: Windows-Startmenü → Einstellungen → Regions- und Sprachoptionen → Karteikarte ”Sprachen” → Schaltfläche ”Details”).

Hier ist die gewünschte Sprache hinzuzufügen. Danach ist das Eingabegebietsschema auch in der Auswahlliste des Tabelleneditorfeldes vorhanden.

Wenn beispielsweise für die dänische Sprache eine besondere Darstellung bezüglich der Schriftart nötig wird, kann dies mithilfe von den Definitionstypen ”Fonts” und ”Fontklassen” erfolgen.

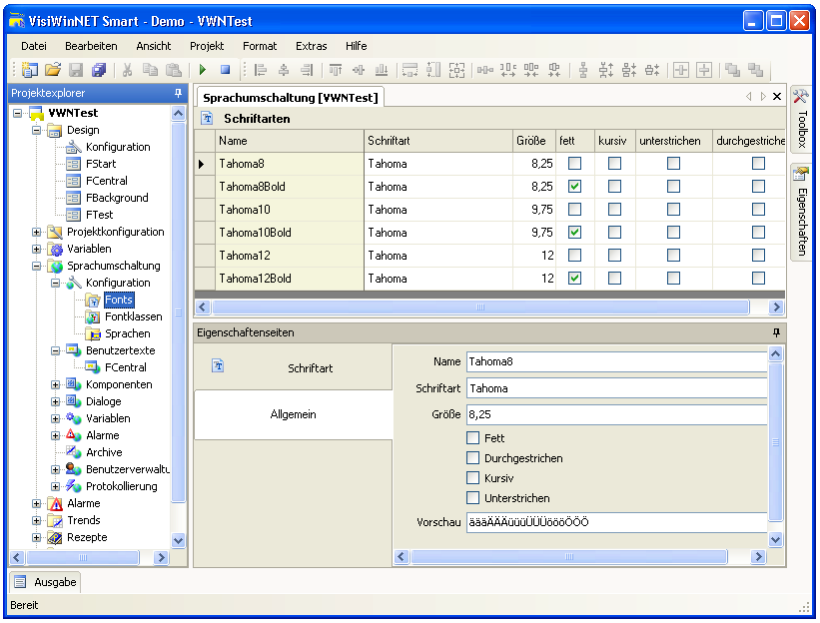
4. Klicken Sie dazu im Projektexplorer doppelt auf den Knoten **Sprachumschaltung** → **Konfiguration** → **Fontklassen**.

   Die Objektliste ”Fontklassen” wird geöffnet, in der Sie für jede Sprache eigene Fontklassen vorgeben können.



Die Fontklassen selbst definieren Sie wiederum über **Sprachumschaltung** → **Konfiguration** → **Fonts**.

Entsprechend welcher Fontklasse der Text eines Steuerelementes dargestellt werden soll, weisen Sie im zugehörigen Eigenschaftenfenster über ”FontClass” zu.

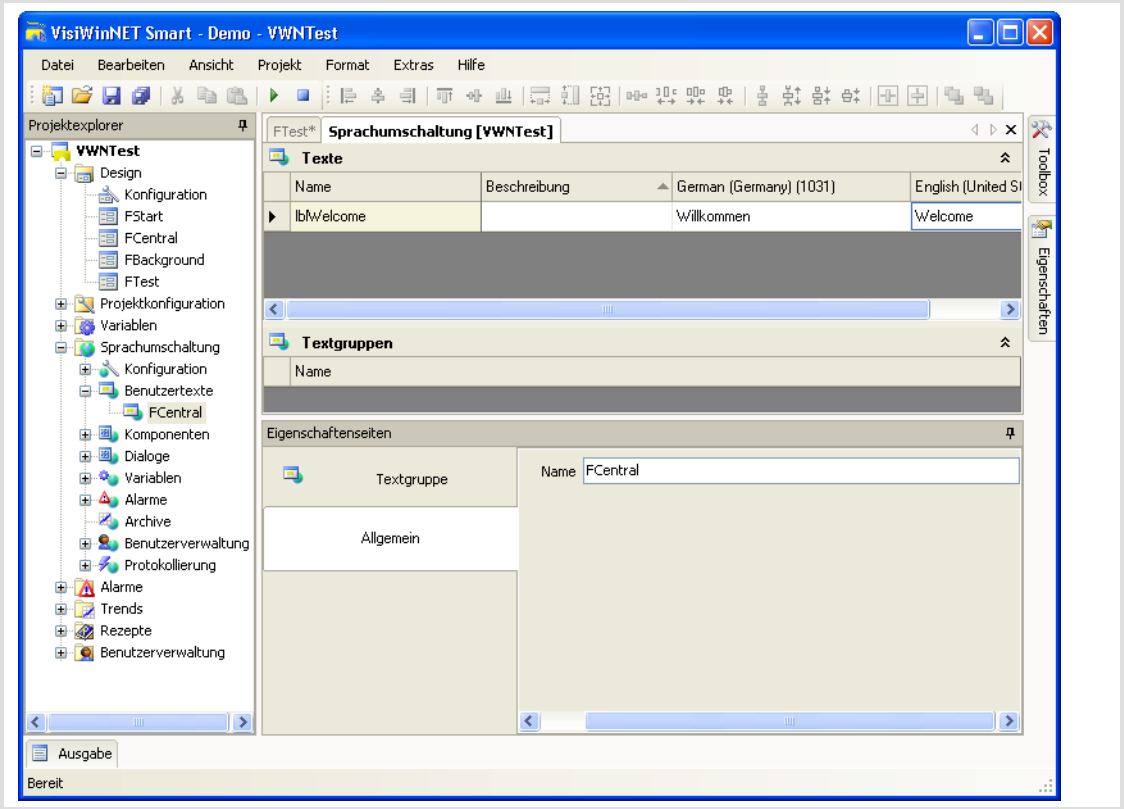


4.6.6 Dynamische Texte verwenden (Variablen anzeigen)

In diesem Abschnitt beschreiben wir, wie Sie Informationen, die zur Laufzeit generiert werden, anzeigen lassen können. Solche dynamischen Informationen sind z. B. Prozessvariablenwerte oder Datums-/Zeitformate.

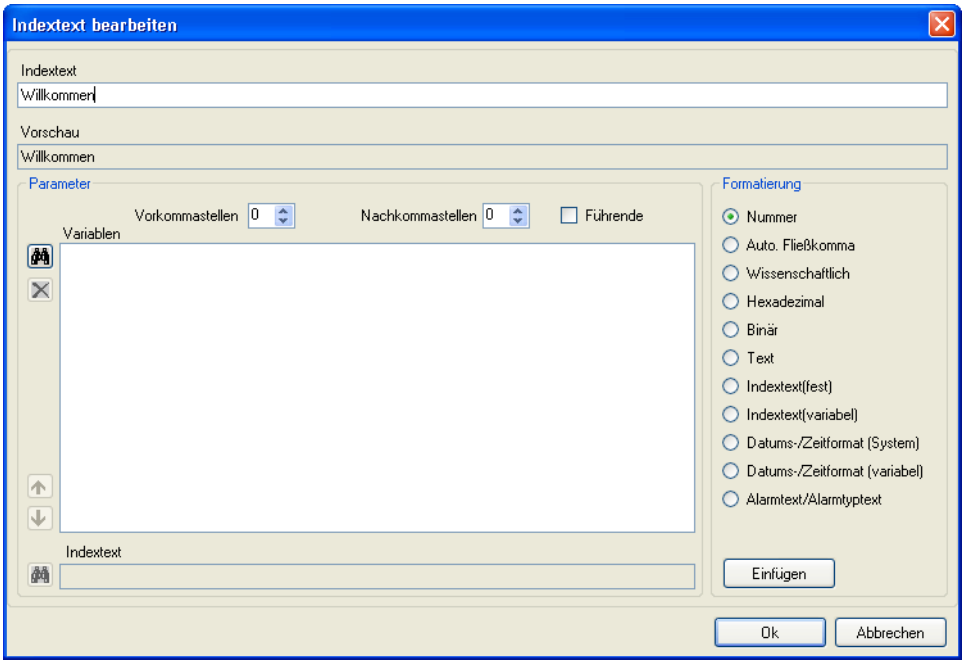
Zur Entwicklungszeit werden durch "@"-Zeichen umfasste Formatzeichenketten in den Text eingefügt. Zur Laufzeit werden diese Bereiche mit den entsprechenden Werten ersetzt.

1. Klicken Sie im Projektexplorer doppelt auf den Knoten **Sprachumschaltung** → **Benutzertexte** → **FCentral**
   Im Clientbereich wird die Objektliste "Texte" und "Textgruppe" angezeigt.

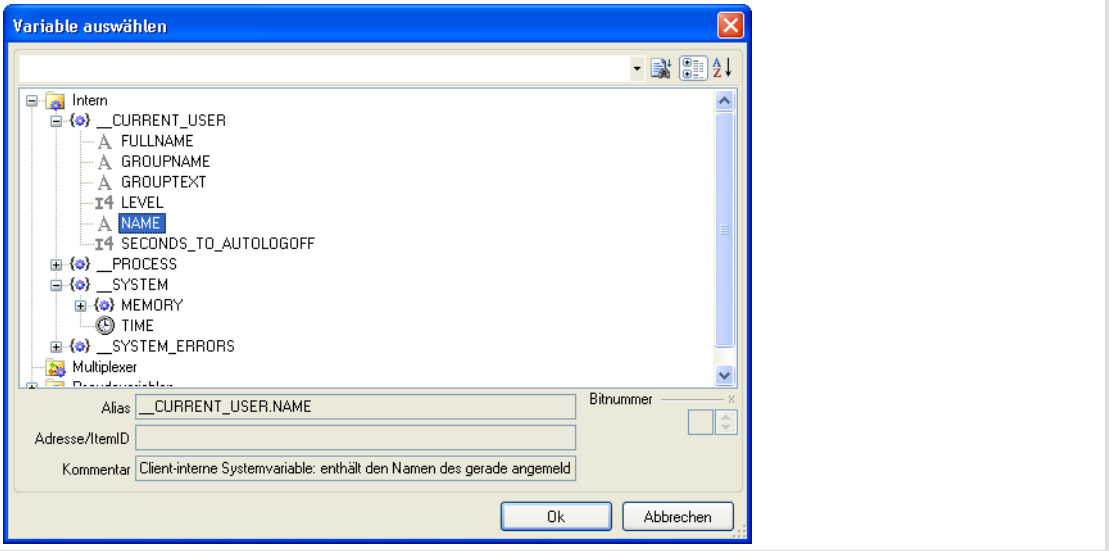


   Sie haben damit Ihren ersten Benutzertext definiert.

2. Markieren Sie das Feld in der Zeile **lblWelcome** und der Spalte **German (Germany)**.
3. Klicken Sie dort auf die Schaltfläche **[...]**.
   Der Dialog "Indextext bearbeiten" erscheint.



4. Klicken Sie in diesem Dialog auf die Schaltfläche mit dem Fernglas, um eine Prozessvariable auszuwählen.



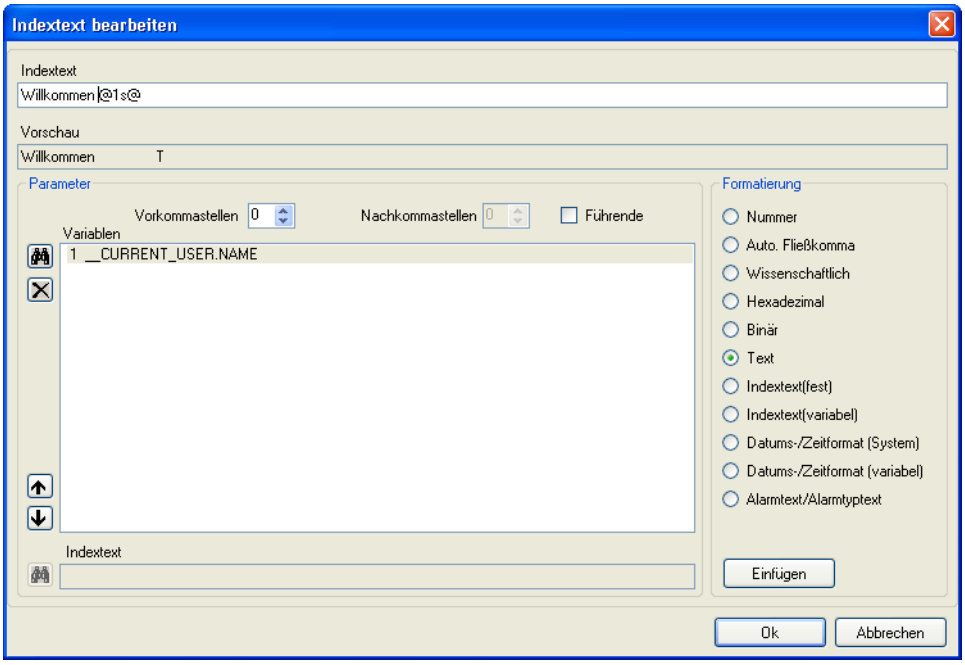
5. Wählen Sie im Dialog **Variable auswählen** unter dem Knoten **Intern** → **__CURRENT_USER** die Variable **NAME** und schließen Sie den Dialog mit **OK**.

   Die Variable ”__CURRENT_USER” wird in das Auswahlfeld im Dialog ”Index bearbeiten” übertragen.

6. Markieren Sie die Variable und das Formatierungs-Optionsfeld **Text**. Klicken Sie anschließend auf **Einfügen**.

   Im Textfeld ”Indextext” steht jetzt: ”Willkommen@1s@”

   Die in ”@”-Zeichen eingefasste Formatierungszeichenkette gibt an, dass der erste in der Prozessvariablenliste angegebene Prozesswert (1) als Text (s) an dieser Stelle ausgegeben werden soll.



7. Fügen Sie noch ein Leerzeichen zwischen ”Willkommen” und ”@1s@” ein und schließen Sie den Dialog über **OK** .
8. Markieren Sie jetzt das Feld in Spalte **English (United States)** und klicken Sie dort auf die Schaltfläche **[...]**.

   Der Dialog ”Indextext bearbeiten” erscheint erneut. Das Listenfeld ist bereits mit der Variablen **__CURRENT_USER.Fullname** vorbelegt.
9. Markieren Sie diese Variable und das Formatierungs-Optionsfeld **Text**. Klicken Sie anschließend auf **Einfügen**.

   Im Textfeld ”Indextext” steht jetzt: ”Welcome @1s@”
10. Fügen Sie noch ein Leerzeichen zwischen ”Welcome” und ”@1s@” ein und schließen Sie den Dialog über **OK** .
11. Starten Sie das Projekt mit **[F5]**.
12. Klicken Sie auf die Schaltfläche **LogOn** und melden Sie sich als Benutzer an (📖 56).
13. Klicken Sie jetzt abwechselnd auf die Schaltflächen **Deutsch** und **Englisch**.

Deutsch	Englisch

4.7 Alarmssystem konfigurieren

Ein Alarmsystem dient zur zentralen Erfassung und Speicherung von Fehlerzuständen oder spezifischen Alarmen. Die entsprechenden Definitionen, welche Variable einen Alarm auslöst, wie der Alarm dargestellt wird und welche Zusatzinformationen zum Alarm gehören, werden im Alarmeditor projektiert. Die Variablen, die die Alarme auslösen, werden von VisiWinNET® Smart beim Variablenkern zur Überwachung angemeldet. Bei einer Variablenwertänderung wird der entsprechende Alarm zur Applikation übertragen, wo sie in speziellen Steuerelementen angezeigt wird.

VisiWinNET® Smart beinhaltet zwei Steuerelemente, die die direkte Anzeige anstehender Alarme erlauben:

AlarmList: Alle anstehenden Alarme werden in Form einer Liste angezeigt. Das Steuerelement unterstützt dabei zusätzlich die Quittierung von Alarmen.

AlarmLine: Die platzsparende Variante, die auf allen Formularen im Kopf- oder Fußbereich Einsicht in anstehende Alarme erlaubt.

Zusätzlich zur Übertragung der Alarmdaten in die Oberfläche erlaubt VisiWinNET® Smart auch die Speicherung von Alarmen in Dateien. Dadurch ist ein Rückblick auf den Produktionsverlauf einer Anlage möglich. Das Steuerelement ”HistoricalAlarmList” dient zur Anzeige dieser aufgezeichneten Daten in Form einer Liste.

Ein Alarm wird durch die Änderung eines Bits aus dem Variablenkern gesteuert. Wählbar ist, ob das Bit auf ”1” oder ”0” gesetzt wird, damit der Alarm ausgelöst wird. Über eine Erweiterung ist es aber auch möglich, analoge Größen beispielsweise auf Grenzwertüberschreitungen zu überprüfen. Alarme können vom Benutzer quittiert werden. Eine Quittierrückmeldung an die Steuerung ist ebenfalls realisierbar.

Das Erscheinungsbild von Alarmen in den Steuerelementen ”AlarmLine” und ”AlarmList” wird über Alarmklassen parametriert. Alarmklassen beinhalten verschiedene Parameter (Farben, Statustexte und Symbole), die die Anzeige eines Alarms beschreiben. Jeder Alarm referenziert eine Alarmklasse. So stehen jedem Alarm die Anzeigeparameter einer Alarmklasse zur Verfügung.

Alarmgruppen gliedern Alarme. Ähnlich wie in der Sprachumschaltung ist es sinnvoll, die einzelnen funktionalen Elemente einer Anlage zu gruppieren. Dies fördert die Übersichtlichkeit und macht das Projekt modularer. Alarmgruppen beinhalten aber auch zusätzlich Parameter, die die gruppenweise Deaktivierung oder Quittierung von Alarmen zulassen.

Das folgende Beispiel beschreibt ...

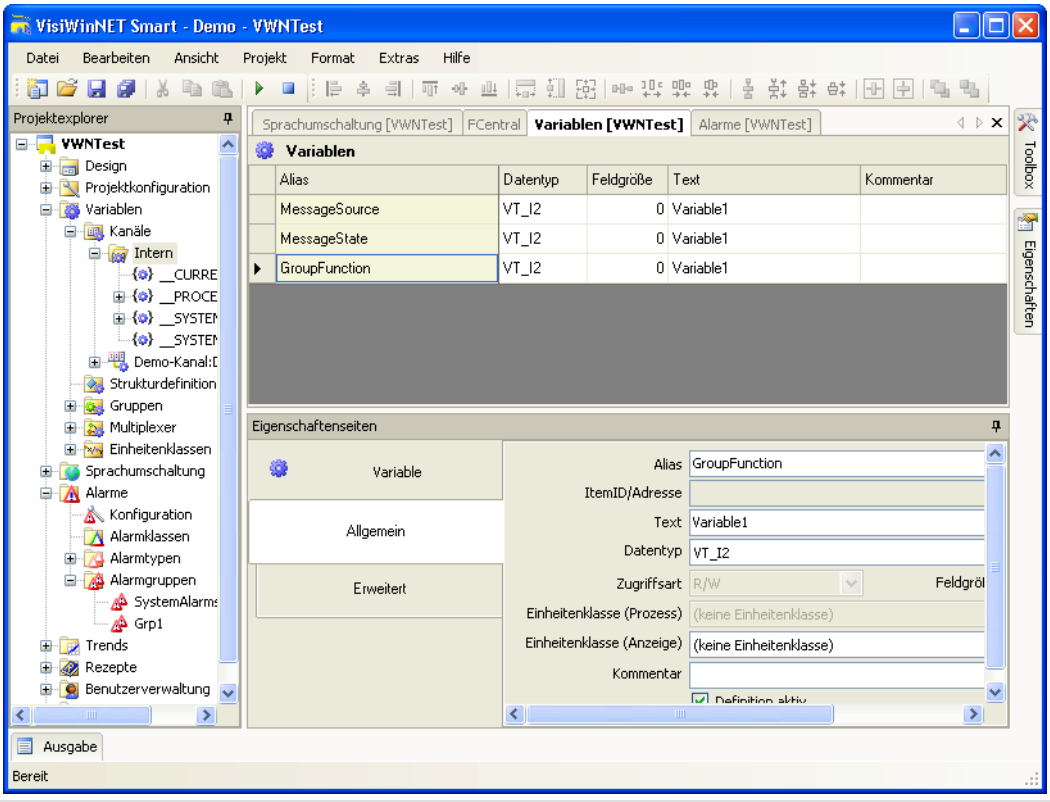
- das generelle Projektieren, Auslösen und Anzeigen von Alarmen, die als Bits im Variablenkern vorliegen.
- das Quittieren von Alarmen.
- verschiedene Anzeigefilter.
- die Überwachung von analogen Werten.

Als Erstes sind die Quellen der Alarme zu definieren. Da Alarme variablengebunden sind, beginnt also alles in der Prozessanbindung. Da allerdings keine realen Steuerungsdaten zur Verfügung stehen, müssen die auslösenden Variablen über interne Variablen simuliert werden.

4.7.1 Prozessvariablen definieren

1. Klicken Sie im Projektexplorer doppelt auf den Knoten **Variablen** → **Kanäle** → **Intern**. Der Tabelleneditor zum Definieren von Variablen wird geöffnet.
2. Definieren Sie über **[F8]** folgende drei Prozessvariablen:

Eigenschaften für Prozessvariablen		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Alias	MessageSource
	Datentyp	VT_I2
2	Allgemein	
	Alias	MessageState
	Datentyp	VT_I2
3	Allgemein	
	Alias	GroupFunction
	Datentyp	VT_I2



Als funktionale Datenanbindung reichen die hier definierten Variablen:·

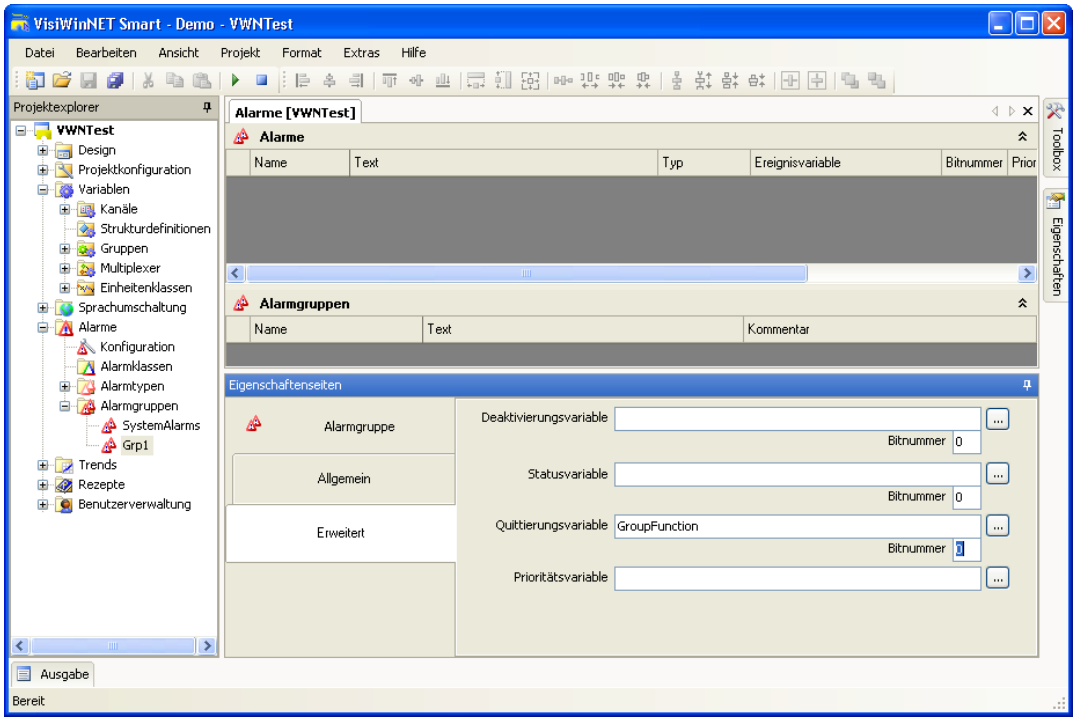
– **MessageSource** beinhaltet die Bits, die die Alarme auslösen.
– **MessageStates** beinhaltet die von der Kernfunktionalität zurückgegebenen Alarmzustände.
– **GroupFunction** dient zur Steuerung von Alarmfunktionalitäten über die übergeordnete Gruppe.

Nun folgt die Festlegung der Definitionen im Alarmsystem.

4.7.2 Alarmgruppen anlegen

1. Klicken Sie im Projektexplorer mit der rechten Maustaste auf den Knoten **Alarme** → **Alarmgruppen** und wählen Sie im Kontextmenü den Befehl **Neu**.
   Der Tabelleneditor zum Definieren von Alarmen und Alarmgruppen wird geöffnet.
2. Legen Sie folgende Eigenschaften fest:

Eigenschaften für Alarmgruppen		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Name	Grp1
	Quittierungsvariable	GroupFunction
	Bit-Nr. Quittierungvariable	0



3. Klicken Sie im Projektexplorer erneut mit der rechten Maustaste auf den Knoten **Alarme** → **Alarmgruppen** und wählen Sie im Kontextmenü den Befehl **Neu**.
   Alarmgruppen sind schachtelbar, daher zuerst zum Knoten ”Alarmgruppen” wechseln!
4. Legen Sie folgende Eigenschaften fest:

Eigenschaften für Alarmgruppen		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Name	Grp2
	Quittierungsvariable	GroupFunction
	Bit-Nr. Quittierungvariable	1

Jetzt benötigen wir noch zwei Alarme für jede Alarmgruppe.

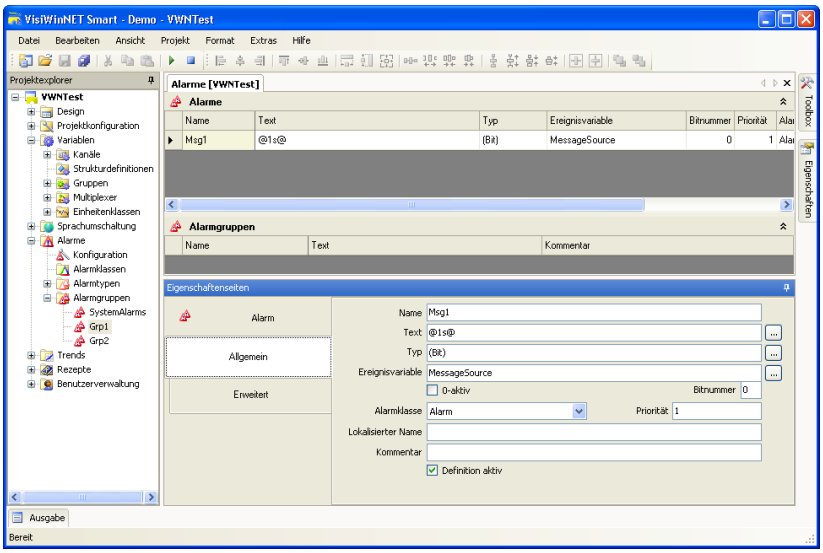
4.7.3 Alarme anlegen

Alarme für Alarmgruppe ”Grp1”

1. Zum Anlegen der beiden Alarme in der Alarmgruppe Grp1, markieren Sie im Projektexplorer die Alarmgruppe **Grp1**, klicken im Clientbereich mit der rechten Maustaste und wählen im Kontextmenü den Befehl **Neu**.
2. Legen Sie folgende Eigenschaften fest:

Eigenschaften für Alarme

Nr.	Register/Eigenschaft	Wert	Bemerkung
1	Allgemein		
	Name	Msg1	
	Text	MessageSource; Bit 0	Auf **[...]** klicken, dann auf die Schaltfläche mit dem **Fernglas**, **Variable MessageSource** wählen und **Bitnummer 0** vorgeben
	Ereignisvariable	MessageSource	
	Bit-Nr. Ereignisvariable	0	
	Priorität	1	
	Erweitert		
	Statusvariable	MessageState	
	Bit-Nr. Statusvariable	0	
2	Allgemein		
	Name	Msg2	
	Text	MessageSource; Bit 1	s. o.
	Ereignisvariable	MessageSource	
	Bit-Nr. Ereignisvariable	1	
	Priorität	2	
	Erweitert		
	Statusvariable	MessageState	
	Bit-Nr. Statusvariable	4	



Alarme für Alarmgruppe ”Grp2”

3. Zum Anlegen der beiden Alarme in der Alarmgruppe Grp2, markieren Sie im Projektexplorer die Alarmgruppe **Grp2**, klicken im Clientbereich mit der rechten Maustaste und wählen im Kontextmenü den Befehl **Neu**.
4. Legen Sie folgende Eigenschaften fest:

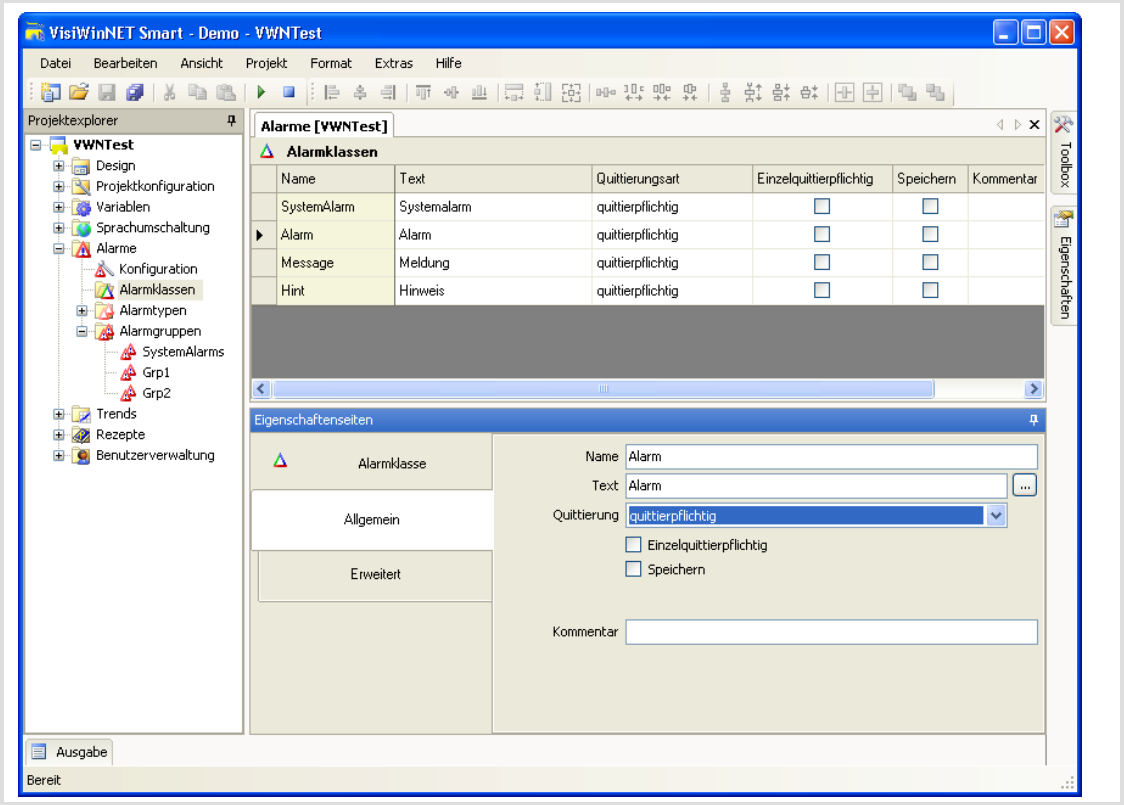
Eigenschaften für Alarme			
Nr.	**Register/Eigenschaft**	**Wert**	**Bemerkung**
3	Allgemein		
	Name	Msg3	
	Text	MessageSource; Bit 2	s. Alarme für Alarmgruppe ”Grp1”
	Ereignisvariable	MessageSource	
	Bit-Nr. Ereignisvariable	2	
	Priorität	3	
	Erweitert		
	Statusvariable	MessageState	
	Bit-Nr. Statusvariable	8	
4	Allgemein		
	Name	Msg4	
	Text	MessageSource; Bit 3	s. Alarme für Alarmgruppe ”Grp1”
	Ereignisvariable	MessageSource	
	Bit-Nr. Ereignisvariable	3	
	Priorität	4	
	Erweitert		
	Statusvariable	MessageState	
	Bit-Nr. Statusvariable	12	

Die oben angegebenen Werte werden im Folgenden die volle Funktion des Alarmsystems zeigen. Der Parameter ”Statusvariable” ist optional. Die Festlegung eines Alarmtextes wird dringend empfohlen. Die Angabe der Ereignisvariable muss sein.

4.7.4 Alarm-Quittierungsart festlegen

Alle neu angelegten Alarme zeigen mit dem Parameter ”Alarmklasse” auf den Wert ”Alarm”. Dahinter steckt eine Definition, die sich unter dem Knoten ”Alarmklassen” des Projektexplorers verbirgt.

1. Klicken Sie im Projektexplorer auf den Knoten **Alarme** → **Alarmklassen**.
   Die Tabelle der Alarmklassen wird angezeigt.
2. Markieren Sie die Zeile **Alarm** und setzen Sie dafür auf der Eigenschaftenseite den Parameter **Quittierung** auf **quittierungspflichtig**.

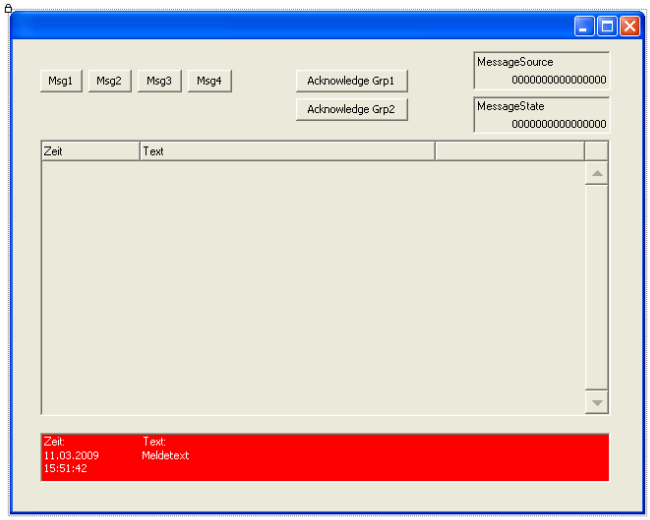


VisiWinNET® Smart unterstützt verschiedene Quittierungsarten. Eine Quittierungsart legt fest, wie oft ein Alarm vom Benutzer oder von der Steuerung quittiert werden muss, bevor sie aus den Steuerelementen der Applikation verschwindet. Durch diese Einstellung kann ein Alarm zur Laufzeit verschiedene Stati einnehmen. Beispielsweise unterscheidet das System zwischen dem Zustand ”gekommen” (Ereignisbit wurde gesetzt) und ”gekommen quittiert” (Nach dem Setzen wurde vom Benutzer das Auftreten quittiert, das Ereignisbit ist aber noch nicht zurückgesetzt worden). Jeder Zustand kann durch ein Symbol, einen Statustext und Farben kenntlich gemacht werden. Auch diese Festlegungen erfolgen in den Alarmklassen.

Der Status eines Alarms kann auch bitcodiert in einer Variablen hinterlegt werden. Dazu sind in den Alarmen die Statusvariablen mit Bitnummern festgelegt worden. Jeweils drei Bits repräsentieren den Status eines Alarms.

4.7.5 Alarme visualsieren

1. Öffnen Sie das Formular **FAlarm** (📖 44) und platzieren Sie dort folgende Steuerelemente:



Eigenschaften für Steuerelemente

Nr.	Steuerelementtyp/Eigenschaft	Wert	Bemerkung
1	CommandButton		
	Darstellung/Text	Msg1	
	VisiWinNET/VwItem	Variable ”MessageSource” mit Bit-Nr. 0	Bit löst den Alarm aus
2	CommandButton		
	Darstellung/Text	Msg2	
	VisiWinNET/VwItem	Variable ”MessageSource” mit Bit-Nr. 1	
3	CommandButton		
	Darstellung/Text	Msg3	
	VisiWinNET/VwItem	Variable ”MessageSource” mit Bit-Nr. 2	
4	CommandButton		
	Darstellung/Text	Msg4	
	VisiWinNET/VwItem	Variable ”MessageSource” mit Bit-Nr. 3	
5	Key		
	Darstellung/Text	Acknowledge Grp1	
	VisiWinNET/VwItem	Variable ”GroupFuction” mit Bit-Nr. 0	
6	Key		
	Darstellung/Text	Acknowledge Grp2	
	VisiWinNET/VwItem	Variable ”GroupFuction” mit Bit-Nr. 1	
7	VarOut		
	Darstellung/Label/Text	MessageSource	
	VisiWinNET/VwItem	Variable ”MessageSource” mit Format ”Binary”	

Eigenschaften für Steuerelemente			
Nr.	**Steuerelementtyp/Eigenschaft**	**Wert**	**Bemerkung**
8	VarOut		
	Darstellung/Label/Text	MessageState	
	VisiWinNET/VwItem	Variable ”MessageState” mit Format ”Binary”	
9	AlarmList		Listenfeld; braucht nicht angepasst werden
10	AlarmLine		rote Zeile unter Listenfeld; braucht nicht angepasst werden

4.7.6 Alarmsystem testen

1. Starten Sie das Projekt über **[F5]**, klicken Sie auf die Schaltfläche **FAlarm** und setzen Sie die vier Schalter **Msg1** ... **Msg4**.
   – In der Alarmliste werden alle vier Alarme angezeigt·
   – In der Alarmzeile wird der Alarm ”Msg4” angezeigt. Dies ist der aktuell im System anstehende Alarm mit der höchsten Priorität.
   – Die Variable ”MessageState” zeigt jeweils für jeden Alarm den Status ”0100” (binär) an. Nötigenfalls könnte dieser Status auch an die Steuerung zurück übertragen werden.
2. Setzen Sie die vier Schalter **Msg1** ... **Msg4** wieder zurück.
   – In der Alarmliste bleiben die Alarme erhalten. Es ändert sich jedoch der Status und damit die Farbe. Durch die gewählte Quittierungsart ist eine Quittierung nötig, um die Alarme endgültig zurückzusetzen.
   – Die Statusbits von Alarmen und Alarmgruppen zeigen jetzt den Status ”0010”.
3. Quittieren Sie die Alarme nacheinander durch einen Doppelklick auf den entsprechenden Eintrag in der Alarmliste.
   – Die Alarme verschwinden aus der Alarmliste.
   – Auch die Statusbits werden in ”MessageState” zurückgesetzt.
4. Setzen Sie die vier Schalter **Msg1** ... **Msg4** erneut und klicken Sie dann auf beide Key-Steuerelemente.
   Über das Quittier-Bit einer Gruppe werden die enthaltenen Alarme gemeinsam quittiert.

4.7.7 Alarm-Anzeigen filtern

VisiWinNET® Smart bietet die Möglichkeit, die Anzeige von Alarmen in den Steuerelementen zu filtern. Dadurch ist es beispielsweise möglich, eine Alarmliste nur mit Alarmen über Anlagenstörungen zu projektieren und eine weitere mit Produktionshinweisen. Im Client-Server-System wird dadurch auch möglich, nur die für den Arbeitsplatz wichtigen Alarme anzuzeigen und trotzdem eine zentrale Datenhaltung zu behalten.

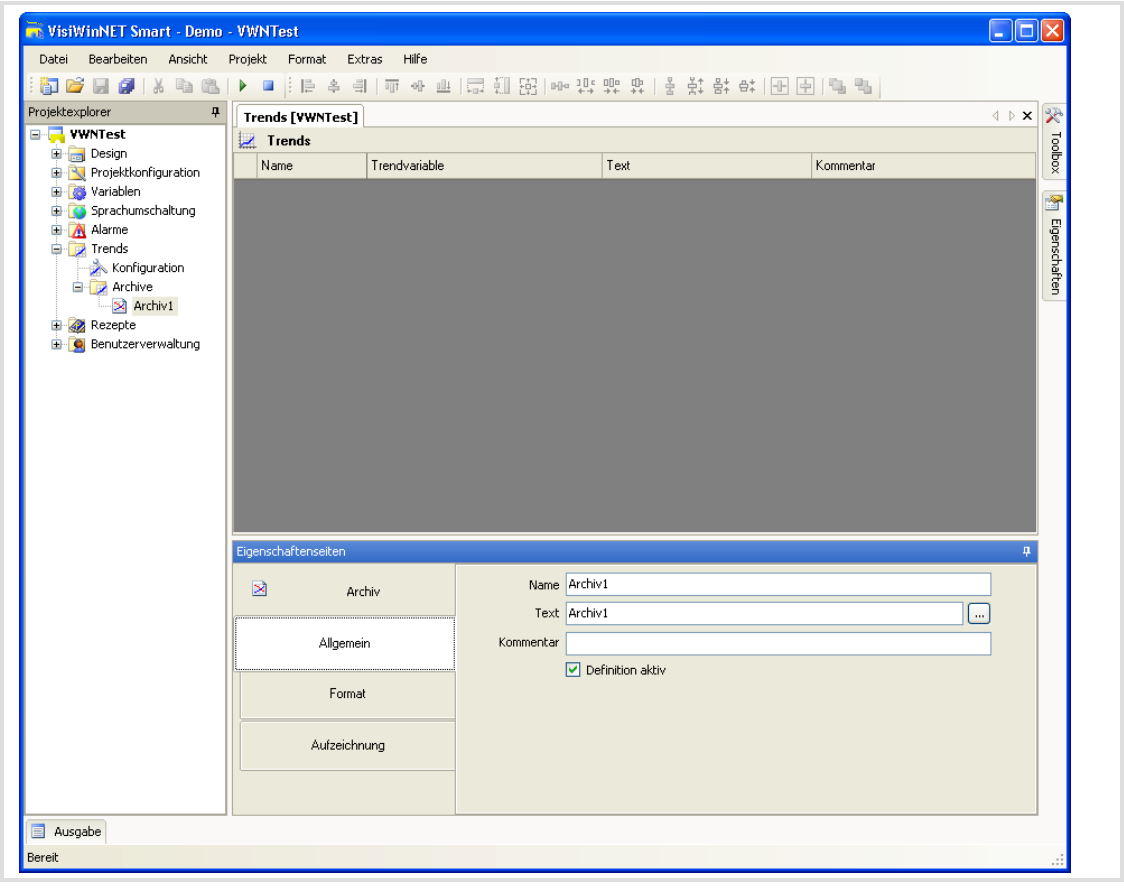
1. Klicken Sie auf dem Steuerelement **AlarmList** rechts oben auf das kleine Dreieck.
   Das Menü "AlarmList Task" wird geöffnet.
2. Klicken Sie dort auf **AlarmFilter (Classes: ; Groups ...)**.
   Der Dialog "AlarmFilterItemEditor" wird geöffnet. Der Dialog ermöglicht die Auswahl von Alarmgruppen und Alarmklassen. Nur die Alarme, die in den hier ausgewählte Gruppen liegen, werden im Steuerelement angezeigt.
3. Führen Sie einen Doppelklick auf dem Eintrag **Grp1** aus.
   Die Alarmgruppe "Grp1" wird ausgewählt.
4. Schließen Sie den Dialog über **OK**.
5. Starten Sie das Projekt über **[F5]** und setzen Sie die vier Schalter **Msg1** ... **Msg4**.
   Nur die in "Grp1" liegenden Alarme werden in der Alarmliste angezeigt. Die Statusbits zeigen aber, dass auch die in "Grp2" liegenden Alarme generiert wurden. Der Filter beschränkt sich also wirklich nur auf die Anzeige im Steuerelement.

4.8 Prozessdaten aufzeichnen (Archivsystem)

Das VisiWinNET® Smart-Archivsystem erlaubt die Aufzeichnung von Prozesswerten. Als Ansichtskomponente steht das ”TrendChart”-Steuerelement zur Verfügung. Die nötigen Definitionen, welche Prozesswerte wie aufgezeichnet werden sollen, werden im Archiv-Editor vorgenommen.

1. Öffnen Sie im Projektexplorer den Knoten **Trends**. Klicken Sie anschließend mit der rechten Maustaste auf den Unterknoten **Archive** und wählen im Kontextmenü den Befehl **Neu**.
2. Definieren Sie auf der Eigenschaftenseite des neuen Archivs folgende Parameter:

Eigenschaften für Archiv		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Name	Archiv1
	Format	
	Archivmodus	Folgearchiv (auf Festplatte)
	Aufzeichnung	
	Abtastzeit	10 ms (Einschränkungen EL 100: 📖 7)



Ein Archiv ist ein Container für aufzuzeichnende Werte. Dieser Container ...·

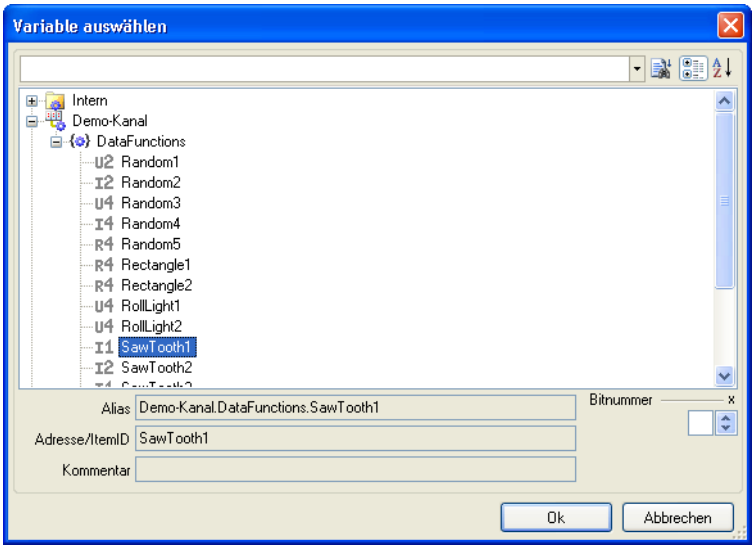
– fasst mehrere aufgezeichnete Prozesswerte in Dateien zusammen
– legt das Abtastverhalten fest
– administriert die erzeugten Dateien

Die Festlegung der aufzuzeichnenden Prozesswerte wird im folgenden Schritt beschrieben.

3. Klicken Sie im Clientbereich mit der rechten Maustaste und wählen Sie den Befehl **Neu**.

   Eine neue Trenddefinition wird angelegt.

4. Öffnen Sie den Variablenauswahl-Dialog über die Schaltfläche **[...]** in der Spalte "Trendvariablen".



5. Wählen Sie eine Variable, z. B. "SawTooth1".

   Damit ist auch die Festlegung des Prozesswertes erledigt. Es fehlt noch die Ansicht der aufgezeichneten Werte in der Applikation.

6. Legen Sie über den Knoten **Design** ein neues Formular **FTrend** an. Weisen Sie dessen Eigenschaften unter **FormBorderStyle** den Wert **Fixed Dialog** zu. (📖 44)

   Das Formular wird damit als Dialog angezeigt, mit den Standard-Windows-Schaltflächen zur Steuerung der Fenstergröße. Sie können darüber den Dialog schließen und zum aufrufenden Fenster wechseln.

7. Erstellen Sie auf dem Formular **FCenter** eine Schaltfläche, die diesen Dialog aufruft (📖 47).

Eigenschaften für Steuerelemente

Nr.	Steuerelementtyp/Eigenschaft	Wert	Bemerkung
1	CommandButton		
	Darstellung/Text	FTrend	
	VisiWinNET/Command	ShowForm(FTrend)	

8. Platzieren Sie auf dem Formular **FTrend** ein Steuerelement vom Typ **Trend Chart** (Sektion "System").

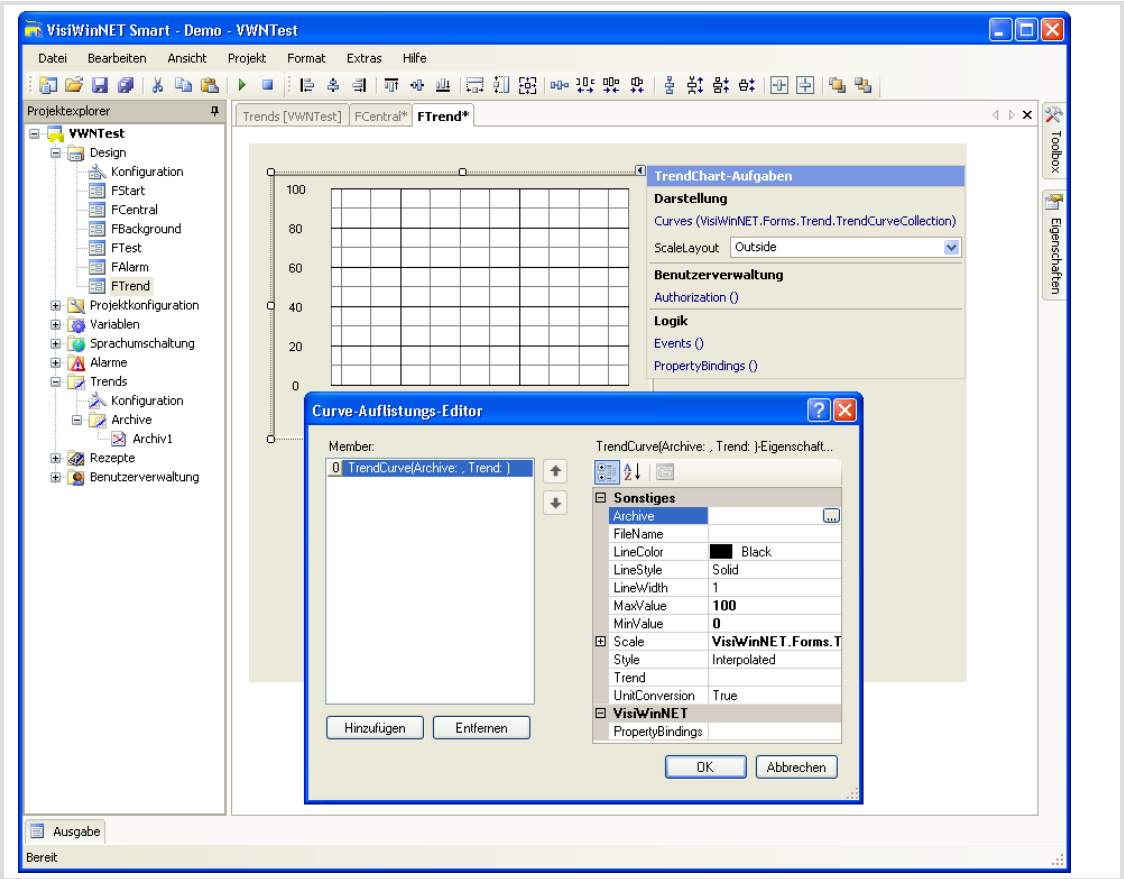
   Wenn Sie das Steuerelement mit einem Mausklick positionieren, müssen Sie es anschließend über seine "Anfasser" größer aufziehen, bis der der Chart seine gewünschte Größe hat.

9. Klicken Sie auf dem sich nun öffnenden Aufgabenmenü "TrendChart-Aufgaben" auf **Curves (VisiWinNet.Forms ...)**.

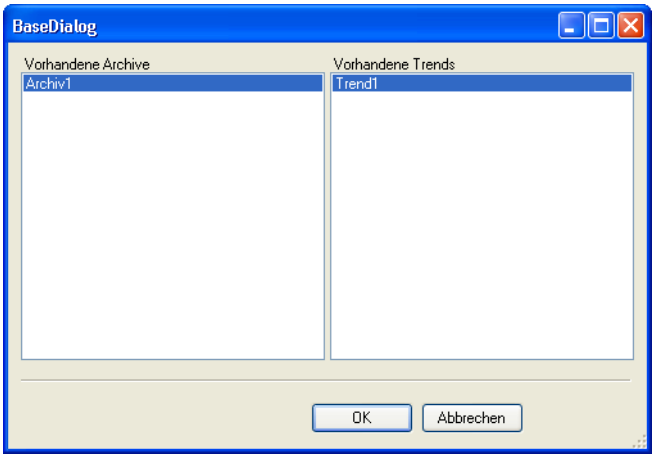
Hinweis!

Das Aufgabenmenü wird geschlossen, sobald Sie außerhalb des Menüs mit der linken Maustaste klicken. Sie können es bei markiertem Steuerelement über die kleine Schaltfläche ▶ am oberen Steuerelement-Rand erneut öffnen.

Der Dialog ”Curve-Auflistungs-Editor” wird geöffnet. Die Liste auf der linken Seite enthält die zu zeichnenden Kurven. Eine Kurve ist schon im Dialog eingefügt, allerdings hat sie noch keine Anbindung an eine Trenddefinition.



10. Markieren Sie den Eintrag auf der linken Seite und klicken Sie auf der rechten Seite neben der ”Archive”-Eigenschaft auf die Schaltfläche **[...]**.



11. Markieren Sie im folgenden Dialog ”BaseDialog” den Eintrag **Archiv1** und **Trend1**. Klicken Sie auf **OK**.
12. Geben Sie in der Eigenschaft **MaxValue** den Wert **150** ein.

    Die Dialogauswahl wird in die Eigenschaften ”Archive” und ”Trend” übernommen. ”MaxValue” parametriert die obere sichtbare Wertegrenze für diese Kurve.
13. Schließen Sie den Dialog ”Curve-Auflistungs-Editor” über **OK** und das Aufgabenmenü über einen Mausklick außerhalb des Menüs.
14. Markieren Sie das Steuerelement ”TrendChart” und klicken Sie oben rechts auf den Karteireiter **Eigenschaften**, um das Eigenschaftenfenster anzuzeigen.
15. Expandieren Sie die Eigenschaft **TimeScale** und geben Sie in der darunter liegenden Eigenschaft **Resolution** den Wert **00:02:00** vor.

    Die ”Resolution”-Eigenschaft legt den sichtbaren zeitlichen Bereich im Steuerelement fest. Hierüber kann also die x-Achse parametriert werden.
16. Starten Sie das Projekt über **[F5]** und wechseln Sie zum Formular **FTrend**.

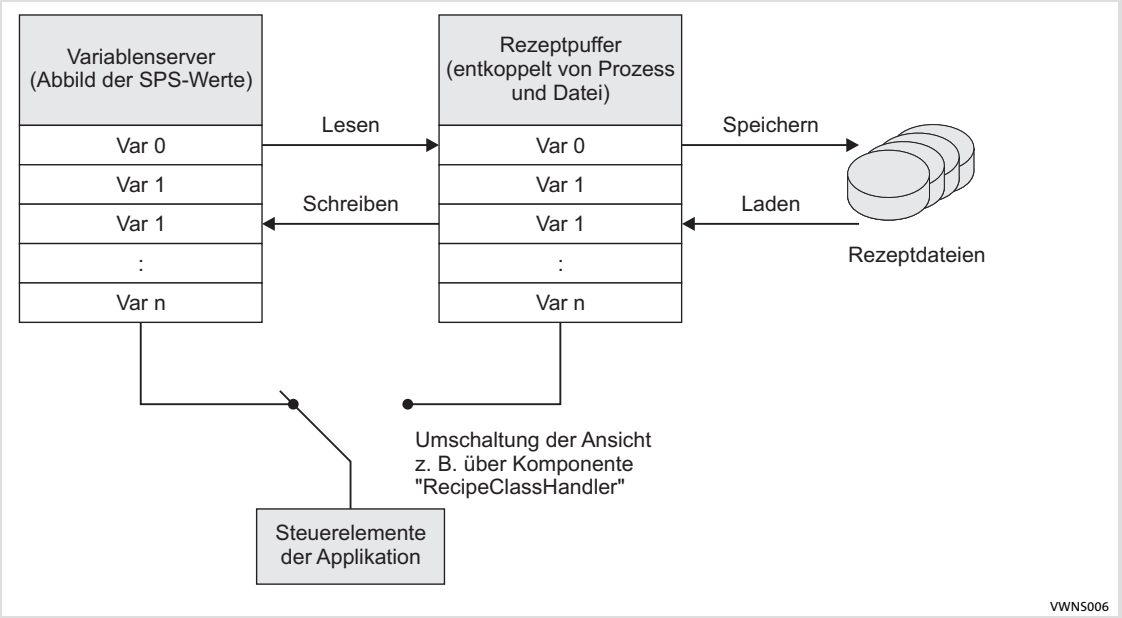
    Die laufende Applikation zeigt den in der Trenddefinition angegebenen Prozesswert als Kurve.

4.9 Visualisierungsrezepte definieren

Visualisierungsrezepte beschreiben Sätze von Prozesswerten. Diese legen üblicherweise gemeinsam die Einstellungen einer Maschine zur Fertigung eines Produktes aus einer Produktpalette fest. Über das Schreiben von Rezepten zur SPS kann damit zwischen den einzelnen zu fertigenden Produkten umgeschaltet werden.

Zur Entwicklungszeit wird über Rezeptdefinitionen festgelegt, welche Variablen die Funktion gemeinsam beschreiben, welche Variablen also gemeinsam gespeichert und transferiert werden.

Zur Laufzeit werden die Prozesswertsätze in sogenannten Rezeptdateien abgelegt. Zwischen dem Prozessabbild und den Rezeptdateien liegt der Rezeptpuffer. Über diesen Puffer können Rezeptwerte in der Applikation geändert werden, ohne dabei SPS-Daten oder Rezeptwertdaten zu beeinflussen. Der Rezeptpuffer dient zur Parametrierung bzw. zur Optimierung von Rezepten. Aus der Applikation werden beispielsweise Rezeptdaten aus einer Datei geladen, im Rezeptpuffer an das Produktionsziel angepasst und anschließend zur SPS transferiert. Der umgekehrte Weg ermöglicht die Kontrolle eingestellter Werte, bevor eine neue Rezeptdatei erzeugt wird.



Die vier Transferfunktionen ”Laden”, ”Speichern”, ”Lesen” und ”Schreiben” steuern die einzelnen Übertragungsschritte.

Typische Eingabesteuerelemente, wie z. B. das ”VarIn”, unterstützen sowohl die Ansicht auf die Prozesswerte im Variablenkern als auch im Rezeptpuffer. Sollen also diese Werte in der Applikation zum Bearbeiten verfügbar gemacht werden, ist die Applikation nicht um neue Steuerelemente zu erweitern. Vielmehr ist die an eine Rezeptdefinition zentral gekoppelte Umschaltung der Ansicht (für alle in der Definition verwendeten Variablen) zu implementieren.

Weitere Funktionen, wie die FDA-konforme Erfassung von Rezeptwertänderungen oder das Speichern von Produktionskommentaren sind einfach parametrierbar.

Die typischen Aufgaben für den Entwickler im Rezeptsystem sind:

Festlegung der Rezeptdefinitionen: Hier wird festgelegt, welche Variablen das Rezept beschreiben. Um allen Irrtümern vorzubeugen: Es geht nicht um die Variablenwerte. Diese werden erst zur Laufzeit festgelegt.

Designanpassung, Editieren der Rezeptwerte: Grundsätzlich können die bestehenden Steuerelemente verwendet werden, die auf die Variablen im Prozess gebunden sind.

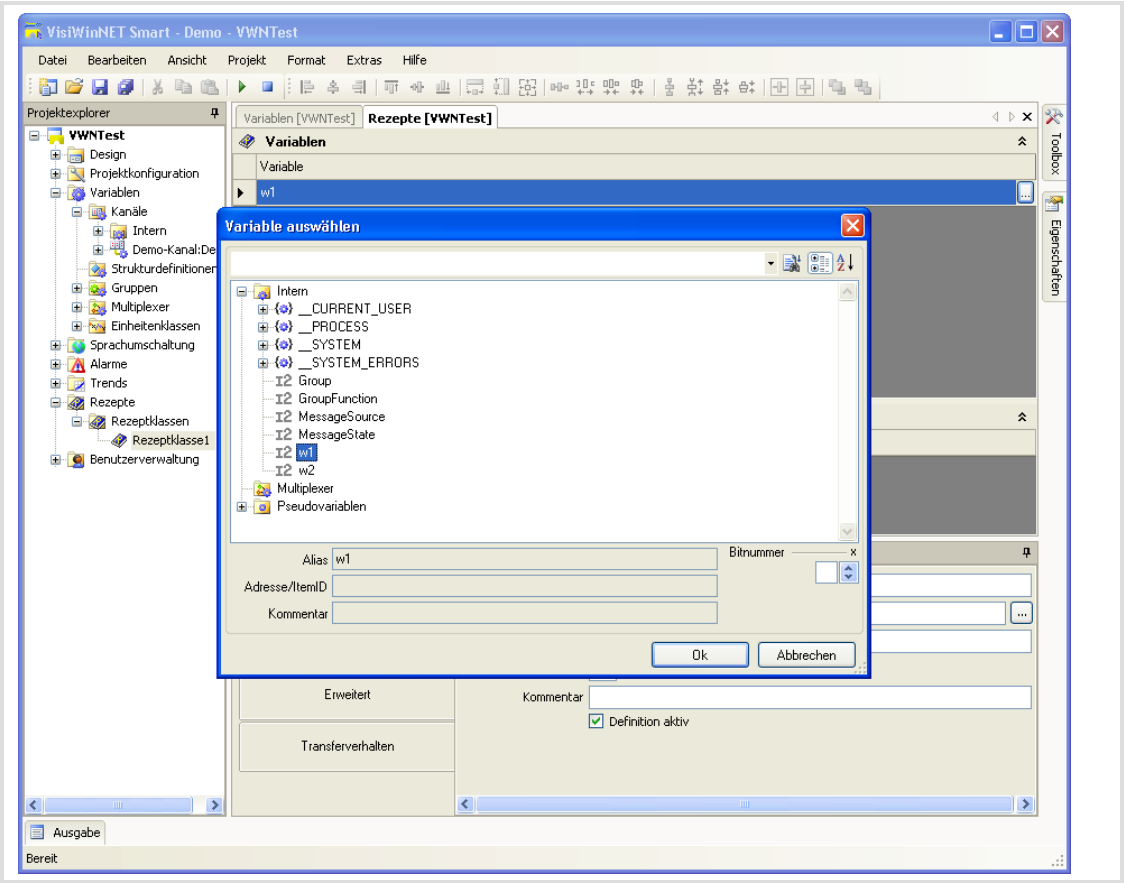
Designanpassung, Administration der Rezeptdateien: Typische Aufgaben, wie das Laden oder Speichern von Rezeptwerten aus Dateien, erfordern meist eine Auflistung der bestehenden Rezeptdateien.

Das folgende Beispiel veranschaulicht die o. g. Schritte anhand eines Minimalrezeptes aus zwei Variablen.

1. Klicken Sie im Projektexplorer doppelt auf den Knoten **Variablen** → **Kanäle** → **Intern**.
   Der Tabelleneditor zum Definieren von Variablen wird geöffnet.
2. Definieren Sie über **[F8]** folgende drei Prozessvariablen:

Eigenschaften für Prozessvariablen		
Nr.	**Register/Eigenschaft**	**Wert**
1	Allgemein	
	Alias	w1
	Datentyp	VT_I2
2	Allgemein	
	Alias	w2
	Datentyp	VT_I2
3	Allgemein	
	Alias	Group
	Datentyp	VT_I2

3. Öffnen Sie im Projektexplorer den Knoten **Rezepte**.
4. Markieren Sie den Unterknoten **Rezeptklassen** und drücken Sie **[F8]**, um eine neue Rezeptklasse anzulegen.
   Der Tabelleneditor wird geöffnet, die ”Rezeptklasse1” ist angelegt.
5. Klicken Sie im Tabelleneditor, im Bereich **Variablen** mit der rechten Maustaste und wählen Sie im Kontextmenü den Befehl **Neu**.
   Ein neues Rezeptelement wird geöffnet.
6. Klicken Sie in der Zeile des neuen Rezeptelements auf die Schaltfläche **[...]**.
   Der Dialog ”Variable auswählen” wird geöffnet. Hier werden die Variablen festgelegt, die zur Rezeptdefinition gehören.



7. Markieren Sie die interne Variable **w1** und schließen Sie den Dialog mit **OK**.

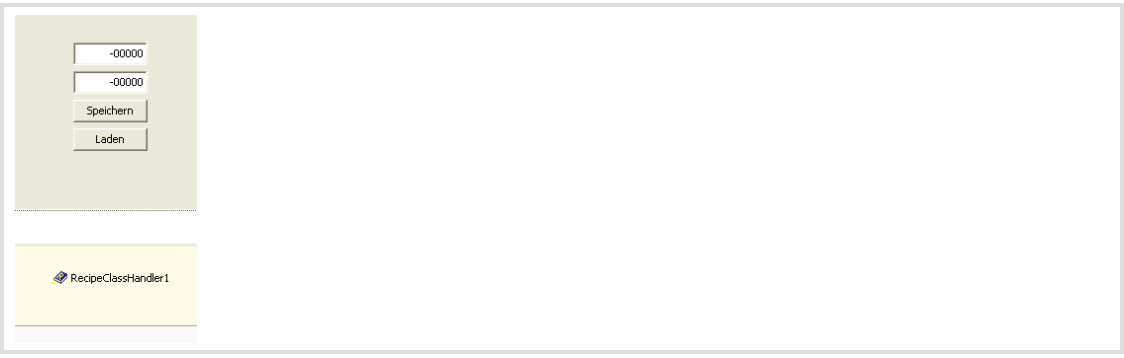
   Die markierte Prozessvariable wird als Element des Rezeptes im Tabelleneditor aufgeführt.

8. Erstellen Sie ein neues Rezeptelement mit der Prozessvariable **w2**wie in den Schritten 5 bis 7 beschrieben.

9. Öffnen Sie im Projektexplorer den Knoten **Design** und öffnen Sie das Formular **FCentral**.

10. Platzieren Sie darauf zwei **VarIn**-Steuerlemente, zwei **CommandButton**-Steuerelemente und ein **RecipeClassHandler**-Steuerelement.

    Nachdem das ”RecipeClassHandler”-Steuerelement auf dem Formular platziert wurde, rutscht es automatisch in das sogenannte ”Komponentenfach” unterhalb der Designansicht des Formulars.



11. Definieren Sie folgende Eigenschaften für die Steuerlemente:

Eigenschaften für Steuerelemente			
Nr.	**Steuerelementtyp/Eigenschaft**	**Wert**	**Bemerkung**
1	VarIn		
	Darstellung/Text	VarIn1	
	VisiWinNET/VwItem	Variable ”w1”	
2	VarIn		
	Darstellung/Text	VarIn2	
	VisiWinNET/VwItem	Variable ”w2”	
3	CommandButton		
	Darstellung/Text	Speichern	
	VisiWinNET/Events	Das Click-Ereignis ist mit dem Aufruf der ”GetRecipe”-Funktion der ”RecipeClassHandler”-Komponente zu verknüpfen. Über die Schaltfläche **[...]** in der ”Events”-Eigenschaft öffnet sich der Dialog ”Ereignisverarbeitung”. Hier ist das **Click**-Ereignis auszuwählen und die Schaltfläche **Auswählen** zu drücken. Auf dem nun sichtbaren Funktionsauswahldialog ist die o. g. Funktion auszuwählen. Nach Schließen des Funktionsauswahldialoges ist sicherzustellen, dass die Funktion in der Funktionenliste erscheint und ihr Parameter ”Provider” auf die ”RecipeClassHandler”-Komponente des Formulars verweist.	
4	CommandButton		
	Darstellung/Text	Laden	
	VisiWinNET/Events	Das Click-Ereignis ist mit dem Aufruf der ”ShowLoadRecipeDialog”-Funktion der ”RecipeClassHandler”-Komponente zu verknüpfen.	
5	RecipeClassHandler		
	VisiWinNET/Events	• Das ”GetDoneSucceeded”-Ereignis ist mit dem Aufruf der ”ShowSaveRecipeDialog”-Funktion der ”RecipeClassHandler”-Komponente zu verknüpfen.· • Das ”LoadDoneSucceeded”-Ereignis ist mit dem Aufruf der ”SetRecipe”-Funktion der ”RecipeClassHandler”-Komponente zu verknüpfen. Die so parametrierten Ereignisse bewirken beim Speichern folgende Abläufe:· • Das Klick-Ereignis triggert die Funktion ”GetRecipe” an. Dadurch werden die Werte der als Rezeptelemente festgelegten Variablen in den Rezeptpuffer geladen.· • Den Abschluss des Einlesens meldet der RecipeClassHandler im ”GetDoneSucceeded”-Ereignis. Dieses wiederum ist mit der ”ShowSaveRecipeDialog”-Funktion verknüpft. Zur Laufzeit wird hierüber das Speichern der Rezeptdatei geregelt, wobei vor dem Speichern ein Dialog zur Eingabe des Namens und der Beschreibung angezeigt wird. Der umgekehrte Weg wird über die Laden-Schaltfläche ausgelöst. Zuerst wird über einen Auswahldialog eine Rezeptdatei in den Rezeptpuffer geladen und anschließend von dort aus in den Prozess transferiert.	
	VisiWinNET/ReceipeClass	Rezeptklasse 1	Der Name der Rezeptdefinition aus der Projektdatenbank.

Dieses Beispiel verwendet den Rezeptpuffer praktisch nicht. In vielen Anwendungen mag es ausreichen, wenn Rezepte direkt im Prozess erstellt und optimiert werden.

Die Verwendung des Rezeptpuffers wird aber dann sinnvoll, wenn Produktionseinstellungen über die Applikation vorzunehmen sind, die während des Einstellens die Werte im Prozess nicht verändern dürfen. Erst nach Abschluss der Einstellungen können die Daten dann gemeinsam in den Prozess übertragen werden.

Im Projekt ”SmartDemo” wird beispielhaft die Verwendung des Rezeptpuffers gezeigt.

4.10 Informationen drucken

VisiWinNET® Smart beinhaltet eine Schnittstelle zum Drucken von Informationen aus der Applikation. Das Layout eines Ausdruckes wird zur Entwicklungszeit über spezielle Druckformulare festgelegt. Der Entwickler platziert Steuerelemente und bestimmt damit die zu druckenden Informationen. Zur Laufzeit füllt das System die Steuerelemente des Formulars mit den entsprechenden Werten. Gestartet wird der Ausdruck über die Komponente "PrintHandler".

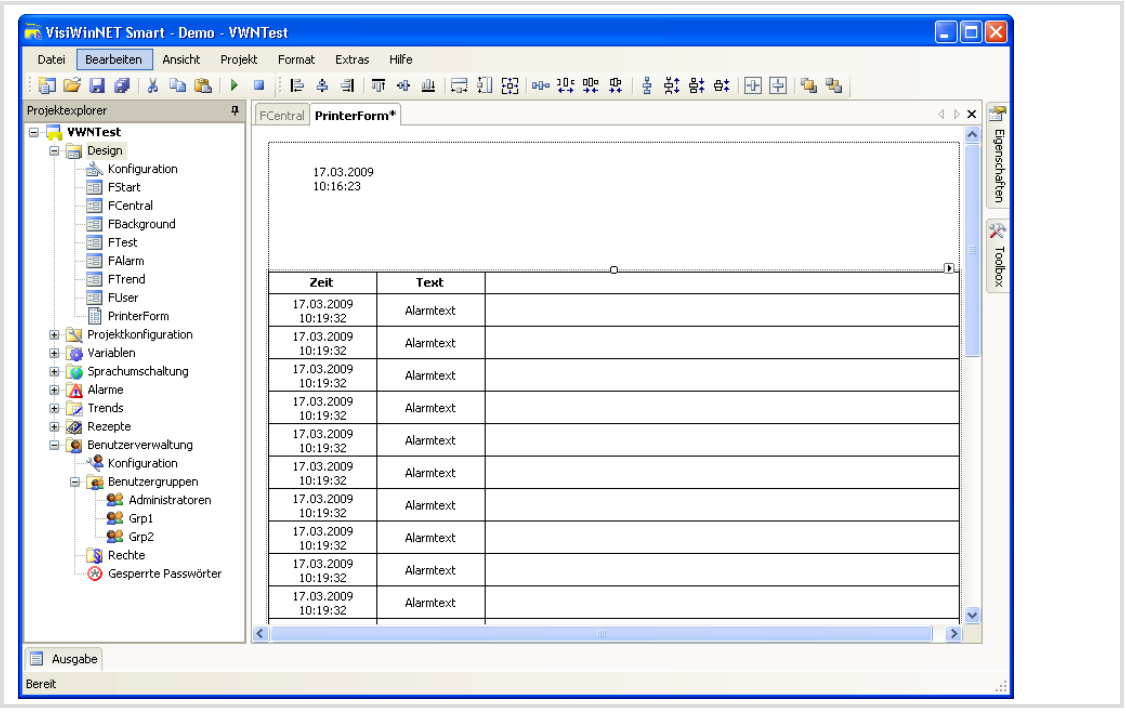
Das folgende Beispiel zeigt die Implementierung einer Druckfunktion in VisiWinNET® Smart.

1. Klicken Sie im Projektexplorer mit der rechten Maustaste auf den Knoten **Design** und wählen Sie im Kontextmenü den Befehl **Druckerformular hinzufügen**.

   Ein Element "PrinterForm" wird unterhalb des Knoten "Design" angelegt und im Clientbereich geöffnet.

   Das Druckerformular-Design entspricht beim Ausdruck einem Blatt Papier. In der Eigenschaft "PageSettings" kann das verwendete Papierformat (Voreinstellung A4) eingestellt werden.

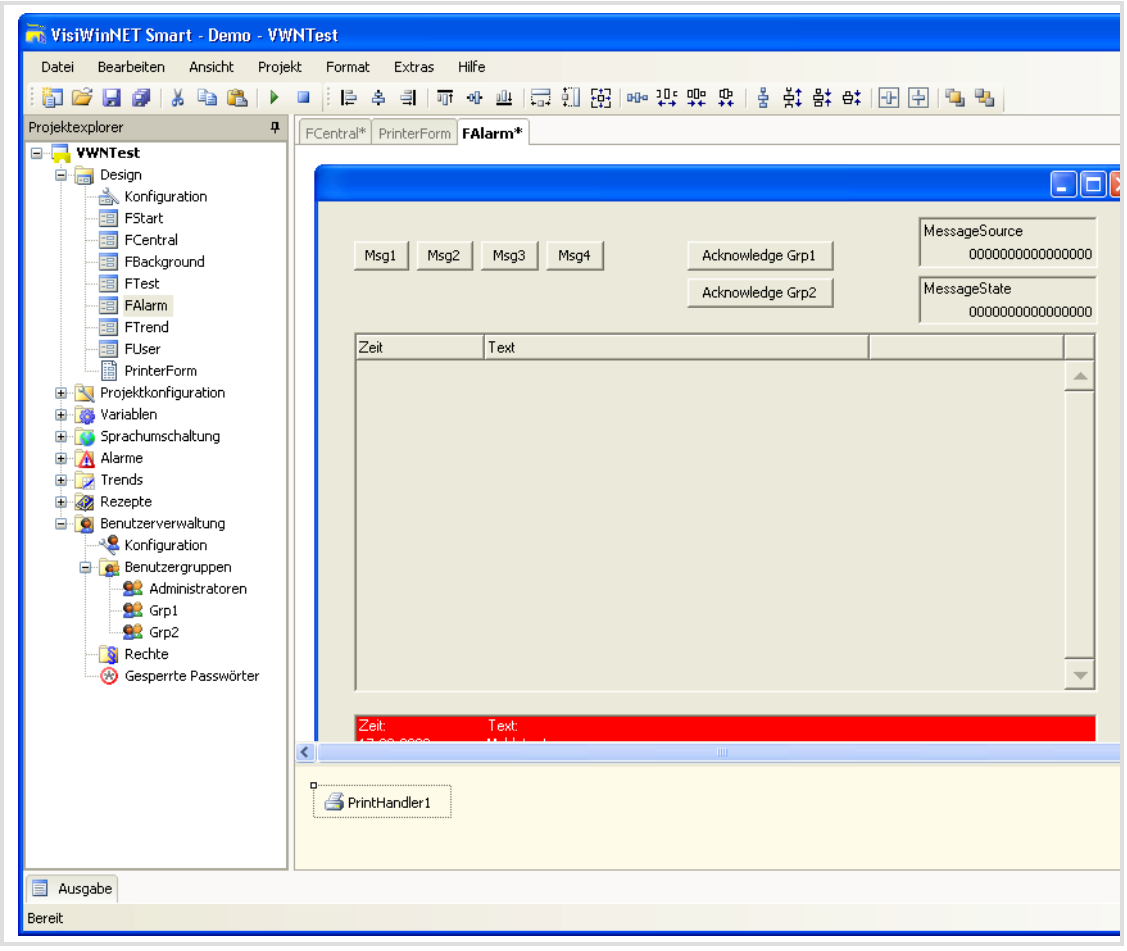
2. Platzieren Sie aus der Toolbox-Sektion "Drucken" ein Steuerelement vom Typ **DateTime** und eines vom Typ **AlarmTable**.



   Mit den zur Verfügung stehenden Steuerelementen lässt sich das Layout und der Inhalt der auszudruckenden Seite festlegen. Um den Ausdruck zur Laufzeit zu starten, sind weitere Arbeitsschritte notwendig.

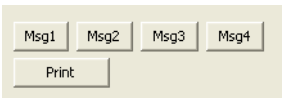
3. Öffnen Sie das Formular **FAlarm** und platzieren Sie dort aus der Toolbox-Sektion **System** ein Steuerelement vom Typ **PrintHandler**

   Nachdem das PrintHandler-Steuerelement auf dem Formular platziert wurde, rutscht es automatisch in das sogenannte "Komponentenfach" unterhalb der Designansicht des Formulars.



4. Weisen Sie diesem Steuerelement in der Eigenschaft **PrinterForms** das angelegte Druckerformular zu: **Eigenschaftenfenster** → **PrinterForms [...]** → **Hinzufügen** → **"PrinterForm"** → **Hinzufügen** → **OK**).
5. Platzieren Sie auf dem Formular ein Steuerelement vom Typ **CommandButton**.
6. Legen Sie auf der Eigenschaftenseite die folgenden Eigenschaften fest:

Eigenschaften für Steuerelemente			
Nr.	**Steuerelementtyp/Eigenschaft**	**Wert**	**Bemerkung**
1	CommandButton		
	Darstellung/Text	Print	
	VisiWinNET/Events	Das Click-Ereignis ist mit dem Aufruf der **Print**-Funktion der **PrintHandler1**-Komponente zu verknüpfen.	



Sie haben damit das Click-Ereignis mit der Print-Funktion der PrintHandler-Komponente verknüpft. Die Komponente wiederum beinhaltet in der PrinterForms-Eigenschaft eine Liste von Druckformularen, deren Inhalte nacheinander in der angegebenen Reihenfolge zum Drucker geschickt werden.

7. Schalten Sie Ihren Standard-Drucker ein und starten Sie das Projekt mit **[F5]**.
   Druckformulare werden immer über den in der Windows-Systemsteuerung festgelegten Standard-Drucker ausgegeben.
8. Wechseln Sie zum Formular **FAlarm**.
9. Aktivieren Sie eine oder mehrere der Schaltflächen **Msgx**.
10. Klicken Sie auf die Schaltfläche **Print**.
    Das Druckformular wird zu Ihrem Standard-Drucker gesendet.

Hinweis!

- Die meisten Windows CE-Geräte unterstützen das Festlegen eines Standard-Drucker nicht. Daher beinhaltet die Komponente ”PrintHandler” die Funktion ”ShowPrinterSettings”. Im Zweifelsfall muss diese Funktion vor dem ersten Druckauftrag einmalig aufgerufen werden, um hier die Druckereinstellungen festzulegen. Das Ergebnis dieser Einstellungen wird im Projektverzeichnis gespeichert. Alle folgenden Druckaufträge werden an den so festgelegten Drucker gesendet.·
- Unter Windows CE sind alle PCL-kompatiblen Drucker verwendbar.

5 Konfiguration

5.1 Projekteinstellungen

VisiWinNET® Smart-Projekte beinhalten Einstellungen, die sich auf das Laufzeitverhalten des Projektes auswirken. Teilweise werden diese Einstellungen schon beim Anlegen eines Projektes angeboten. Andere Einstellungen können nachträglich in der Projektkonfiguration geändert werden.

Die Einstellungen der Projektkonfiguration sind über den gleichnamigen Knoten im Projektexplorer zu erreichen. Der Projektexplorer gliedert die Eigenschaften in verschiedene Bereiche. Nach einem Klick auf einen Bereich werden die entsprechenden Einstellungen in der VisiWinNET-Eigenschaftenseite angezeigt.

Die zur Verfügung stehenden Einstellungen sind abhängig vom Typ des Projektes.

5.1.1 Globale Projekteinstellungen

Zugriff: Projektkonfiguration → Global

Karteikarte auf Eigenschaftenseite: Allgemein

Die allgemeinen Projekteigenschaften geben die Einstellungen wieder, die im Projektmanager beim Anlegen des Projektes vorgenommen wurden.

Eine Änderung der hier angezeigten Einstellungen ist nicht möglich. Folgende Standardaufgabe wird von anderen Teilen der Projektverwaltung vorgenommen:

► Umbenennen des Projektes: Diese Funktion steht im Projektmanager zur Verfügung.

5.1.2 Allgemeine Projekteigenschaften

Zugriff: Projektkonfiguration → Global

Karteikarte auf Eigenschaftenseite: Erweitert

Die erweiterten Eigenschaften geben Ihnen die Möglichkeit, allgemeine Informationen in der Projektdatenbank zu speichern.

5.1.3 Auswahl der Systeme

Zugriff: Projektkonfiguration → Serverkomponenten

Karteikarte auf Eigenschaftenseite: Allgemein

Die hier markierten Systeme werden in der Entwicklungsumgebung als Editoren dargestellt. Mithilfe der Editoren werden die systemspezifischen Definitionen in die Projektdatenbank geschrieben.

Zur Laufzeit werden nur die hier markierten Systemserver geladen. D. h. bei der Abwahl des Trendsystems werden die Aufzeichnungsfunktionen des Trendservers z. B. nicht aktiv.

5.1.4 Erweiterte Projekteigenschaften

Zugriff: Projektkonfiguration → Client (Standard-Projekttyp) oder Projektkonfiguration → Laufzeit (Compact-Projekttyp)

Karteikarte auf Eigenschaftenseite: Kompatibilität

Diese Projekteigenschaft beinhaltet Einstellungen, die ein geändertes Verhalten gegenüber den Versionen 5.x und 6.0 beschreiben.

Einstellung	Beschreibung
Ereignis ”Limit” im Steuerelement ”VarIn” erweitert	Diese Einstellung bezieht sich auf das Auftreten des ”Limit”-Ereignis im ”VarIn”-Steuerelement. Das Ereignis tritt während der Eingabe im Steuerelement auf, wenn eine Grenzwertüberwachung über die Eigenschaften ”LimitCheck” oder ”UseItemLimits” aktiv ist. Insbesondere bei der Überprüfung mehrstelliger Eingabewerte ist zwischen zwei Verhalten zu wählen: ● **deaktiviert** (kompatibel zu Versionen 6.0, 5.x): Das ”Limit”-Ereignis tritt sofort während der Eingabe auf, wenn der angegebene Wert den oberen Grenzwert überschreitet. Unterschreitet der Wert die untere Grenze, wird das ”Limit”-Ereignis so lange zurückgehalten, bis die Eingabe durch Fokuswechsel oder Enter-Taste als abgeschlossen markiert wird. Ist die untere Grenze z. B. auf ”20” gesetzt, kann der Benutzer den Wert ”23” durch das Eintippen der Ziffern ”2” und ”3” ohne Auslösen des ”Limit”-Ereignisses durchführen. ● **aktiviert** (Standardverhalten ab der Version 6.1): Ist die untere Grenze z. B. auf ”20” gesetzt kann der Benutzer den Wert ”23” durch das Eintippen der Ziffern ”2” und ”3” nur noch über das Auslösen des ”Limit”-Ereignisses durchführen. Das Ereignis tritt nach Eingabe der Ziffer ”2” auf, da dieser Wert den unteren Grenzwert unterschreitet. Praktisch angewendet wird diese Einstellung z. B. von der Zifferneingabe der Bildschirmtastatur (NumPad). Hier wird in den Grenzwertfeldern eine unvollständige oder die Grenzen überschreitende Eingabe farblich signalisiert.
Eigenschaft ”DecPoint” im VisiWinNET-Item beschreibbar	Diese Einstellung bezieht sich auf alle VisiWinNET-Steuerelemente, die als ”Prozessgröße” in der VWItem-Eigenschaft eine analoge Größe anbinden. Bei der Anbindung analoger Größen steht innerhalb der Eigenschaft ”VWItem” die Eigenschaft ”DecPoint” zur Verfügung. Die Funktion dieser Eigenschaft wird über die Einstellung ”Eigenschaft ’DecPoint’ im VisiWinNET-Item beschreibbar” beeinflusst: ● **deaktiviert** (kompatibel zu Versionen 6.0, 5.x): Ist im Steuerelement die Eigenschaft ”UnitConversion” auf TRUE eingestellt, hat ”DecPoint”zur Laufzeit keine Funktion. ● **aktiviert** (Standardverhalten ab der Version 6.1): Die durch ”DecPoint” angegebene Kommaverschiebung wird ggf. zusätzlich auf die durch ’UnitConversion’ aktivierte Einheitenumrechnung angewendet. Damit kann zusätzlich zur Umrechnung über eine Einheitenklasse die Angabe von Nachkommastellen bei Fließkommazahlen erfolgen.
Optimierte Aktivierung von Variablen beim Formularwechsel	Auf Formularen benötigte Variablen werden gesammelt beim Variablenkern angemeldet, wenn das Formular vollständig geladen ist. Dies hat eine Geschwindigkeitssteigerung beim Seitenwechsel zur Folge, da auch das Aktivieren benötigter (Deaktivieren nicht mehr benötigter) Variablen beim zugehörigen Kommunikationskanal nun gesammelt und daher schneller erfolgt. Das Aktivieren dieser Option hat jedoch auch zur Folge, dass im ersten ”Change”-Ereignbis eines Items (cause: ChangedByConfig) nicht immer der aktuelle Wert aus dem Kommunikationskanal steht, sondern eventuell ein älterer oder der Startwert. Die Deaktivierung dieser Einstellung schränkt die Funktion folgender Komponenten ein: ● VisiWinNET.DataAccess.MultiItemActivator (Implementiert in VisiWinNET.Standard.Client/ VisiWinNET.Compact.Systems) ● VisiWinNET.Forms.BaseForm (Implementiert in VisiWinNET.Standard.Forms/ VisiWinNET.Compact.Forms) ● In VisiWinNET SMART erben die Formulare direkt von ”VisiWinNET.Forms.BaseForm”. Auch hier hat die Deaktivierung der Einstellung eine funktionale Einschränkung zur Folge.

5.1.5 Laufzeitverhalten

Dieser Konfigurationsdialog ist nur in den ”Standard”-Projekttypen enthalten.

Zugriff: Projektkonfiguration → Laufzeit

Karteikarte auf Eigenschaftenseite: Standard

Beinhaltet Einstellungen der Laufzeit und des VWNManagers

Einstellung	Beschreibung
Zeiteinstellungen	Zykluszeiteinstellungen für die VWEngine. Es kann sowohl die Zykluszeit selbst eingestellt werden (sind alle Daten für einen Zyklus schneller eingetroffen als gewünscht, pausiert der Systemzyklus solange, bis die eingestellte Zykluszeit erreicht ist; dauert die Datenerfassung länger als eingestellt, verlängert sich der Systemtakt automatisch, d.h. keine Datenanforderung geht verloren) als auch die Pausenzeit zwischen zwei Zyklen (dient zur Rechenzeitabgabe an die Visualisierungsapplikation oder andere Prozesse). **ms Zykluszeit:** Die Systemzykluszeit für die Datenerfassung, als Voreinstellung ist ein Grundtakt von 1 Sek. (100 ms) eingestellt. **ms Pausenzeit:** Die Pausenzeit zwischen zwei Systemzyklen (Voreinstellung 1 ms)
VisiWinNET Manager	Das Hauptmenü des VisiWinNET Manager kann konfiguriert werden. **Konfiguration deaktiviert:** Der Dialog Konfiguration kann zur Laufzeit nicht angewählt werden. **Variablendialog deaktiviert:** Der Dialog VWEngine kann zur Laufzeit nicht angewählt werden. **Start/Stop/Beenden deaktiviert:** Die Menüpunkte Start, Stop und Exit können zur Laufzeit nicht angewählt werden.
Laufzeitdaten	**Daten verschlüsseln:** Legt fest, ob die zur Laufzeit erzeugten Daten aus den Systemen ”Protokoll”, ”Rezept” und ”Alarm” verschlüsselt werden sollen. Diese Einstellung bietet eine erhöhte Manipulationssicherheit der erzeugten Daten. **Passwort für Schreibzugriffe:** Legt das Passwort für die zur Laufzeit erzeugten Access-Datenbanken aus den Systemen ”Protokoll”, ”Rezept” und ”Alarm” fest. Wird beim Öffnen der Datenbank (z.B. mit Access) nicht das hier festgelegte Passwort angegeben, kann die Datei nur schreibgeschützt geöffnet werden.
Kommunikation	**Wartezeit auf VisiWin-Treiber:** Gibt die Zeit (in Sekunden) an, die der Variablenkern in der Initialisierungsphase auf einen VisiWin-Treiber wartet. Startet der Treiber in dem hier angegebenen Zeitraum nicht, wird eine Fehlermeldung generiert. **Alle Variablen beim Start 1x lesen:** Gibt an, ob der Variablenkern während der Initialisierungsphase alle Prozesswerte einmal liest. Dieses einmalige Lesen dient dazu, den internen Variablenpuffer zu initialisieren.

5.1.6 Spracheinstellungen

Zugriff: Projektkonfiguration →Global

Karteikarte auf Eigenschaftenseite: Sprachen

Der Dialog listet die im Projekt enthaltenen Sprachen auf. Eine der hier angezeigten Sprachen kann als Entwicklersprache markiert werden. Die Entwicklersprache hat folgende Funktionen:

- In den Formularen verwendete sprachumschaltbare Texte werden zur Entwicklungszeit in der Entwicklersprache angezeigt
- Sprachumschaltbare Texte, wie z. B. ein Alarmtext, werden in den jeweiligen Editoren bei der Eingabe der Entwicklersprache zugeordnet.

5.2 Kommunikation

Zum Lieferumgang von VisiWinNET® Smart gehören diverse Beispielprojekte, die während der Programminstallation auf Ihre Festplatte kopiert wurden.

Neben dem Kennenlernen von VisiWinNET® Smart können Sie diese Projekte sehr gut dazu verwenden, die Verbindung PC ↔ Bedieneinheit EL1xx sowie die Betriebsbereitschaft der Bedieneinheit zu überprüfen.

Um z. B. einen Servo-Umrichter 9400 mit dem EL1xx und dem Beispiel-Projekt steuern zu können, bereiten Sie Folgendes vor:

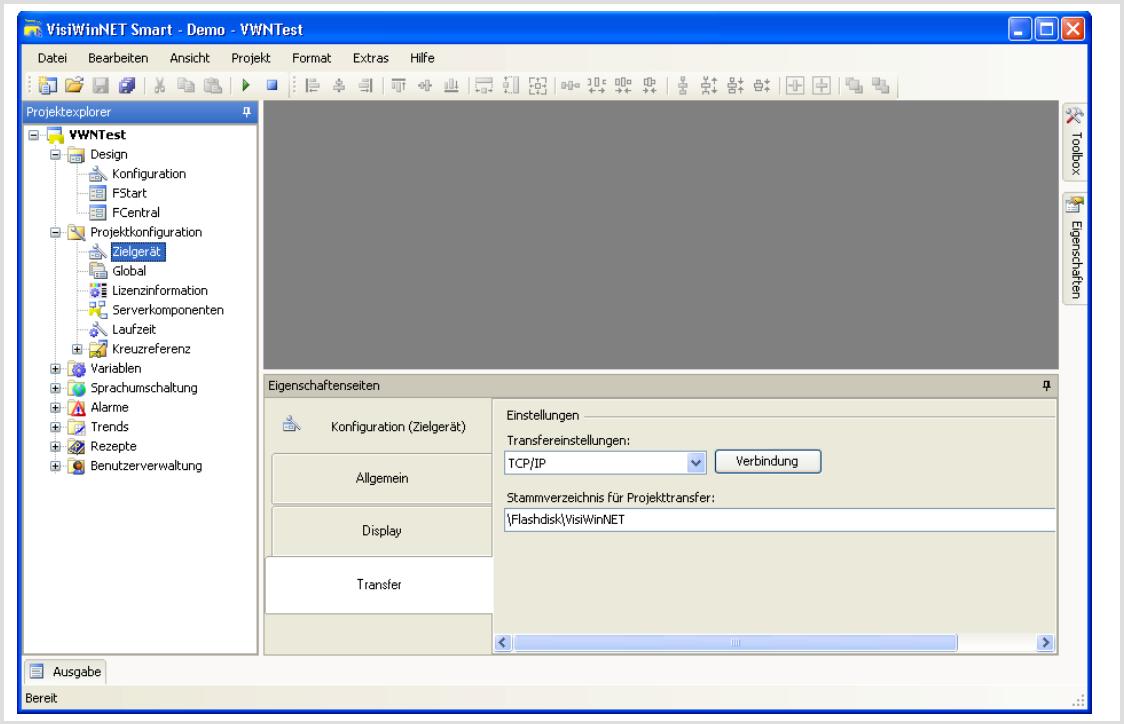
Antriebsgerät	Was ist zu tun	Vorgehensweise
Servo-Umrichter 9400	• Grundkonfiguration vornehmen und die Applikation "Stellantrieb - Drehzahl" wählen • CAN-Knotenadresse 2 vergeben (C0350) • Projekt in den Antriebsregler übertragen • Regler freigeben und Festsollwert 1 aktivieren	🕮 Online-Handbuch Engineer

5.2.1 Projekt über LAN auf das EL1xx übertragen

Voraussetzung: Die Netzwerk-Verbindung zwischen PC und EL1xx ist eingerichtet und aktiv (🕮 Bedienungsanleitung zum HMI EL1xx, Kapitel Ethernet-Verbindung einrichten). Eine auf dem Zielgerät laufende Applikation muss beendet sein.

So gehen Sie vor:

1. Öffnen Sie im Projektexplorer den Ordner **Projektkonfiguration** und anschließend den Unterordner **Zielgerät**.
   Die Eigenschaftenseite "Konfiguration (Zielgerät)" wird angezeigt.
2. Öffnen Sie das Register **Transfer**.



3. Wählen Sie im Auswahlfeld **Transfereinstellungen** die TC/IP-Verbindung.
   Zum Ändern der Verbindungsparameter klicken Sie auf **Verbindung**. Die IP-Adress des EL1xx wird angezeigt, wenn Sie am EL1xx doppelt auf das Netzwerksymbol in der Statuszeile klicken.
4. Geben Sie im Feld ”Stammverzeichnis für Projekt-Transfer” das Download-Ziel für das Laufzeitprogramm auf dem EL1xx vor. Zum Beispiel: \Flashdisk\VisiWinNet. Mögliche Downloadziele:
   \Flashdisk\xxxx: Flashdisk (Programm remanent)
   \Temp: RAM (Programm flüchtig)
   \Storage\xxxx: SD-Karte (Programm remanent und portierbar; wird nur angeboten, wenn am EL 1xx eine SD-Karte gesteckt ist)
   \Hard Disk\xxxx: USB-Speicher-Medium (wie SD-Karte)
5. Klicken Sie in der Symbolleiste auf die Schaltfläche ▶ (”Projekt starten”) und folgen Sie den Anweisungen der Dialoge.
   Ein Dialog wird angezeigt, in dem Sie die zuvor vergebene IP-Adresse editieren können.
6. Klicken Sie auf **Übernehmen**.
   Das Projekt wird auf das Zielgerät übertragen.

5.2.2 CAN-Verbindung konfigurieren über Codestellen

Über die CAN-Schnittstelle kann ein ”EL1xx CAN” mit einem CAN-Feldbussystem kommunizieren.

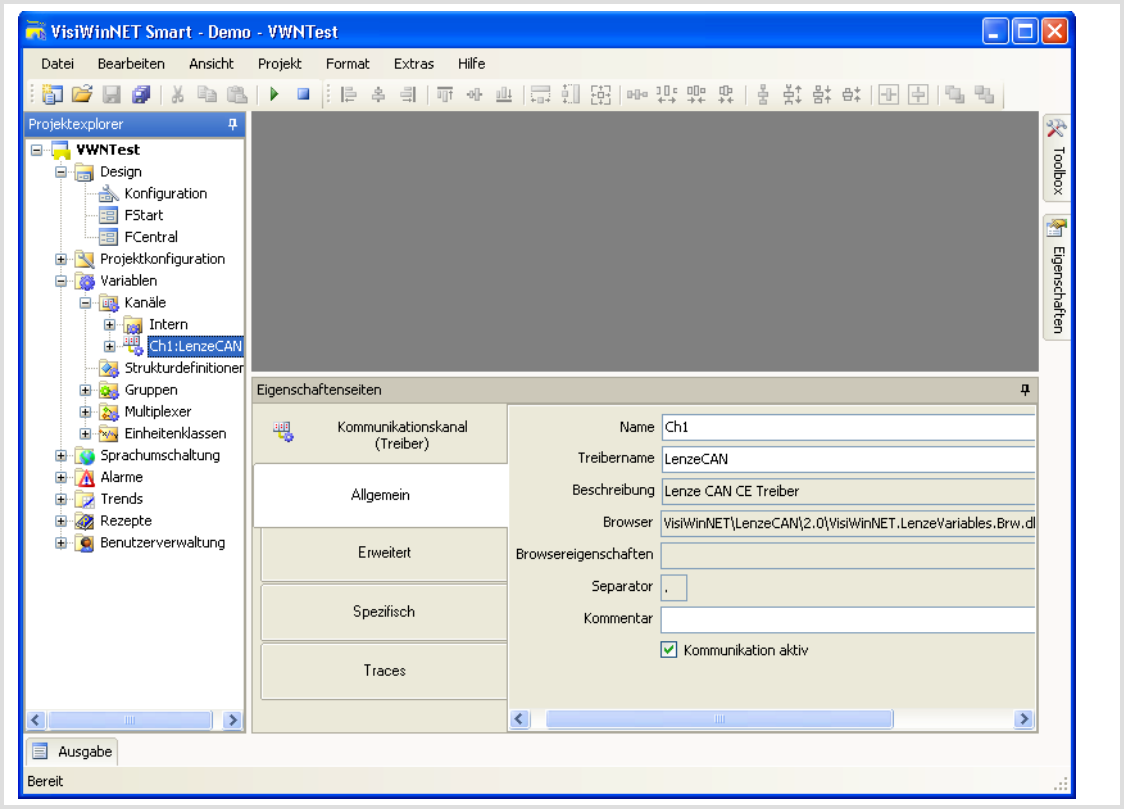
Kanal konfigurieren

Voraussetzung:

- ► Das EL1xx ist über LAN mit dem PC, auf dem VisiWinNET® Smart installiert ist, verbunden.
- ► Das EL1xx ist über CAN mit dem Antriebsregler verbunden.
  Entsprechende Informationen entnehmen Sie der Dokumentation zum EL1xx bzw. zum Antriebsregler.

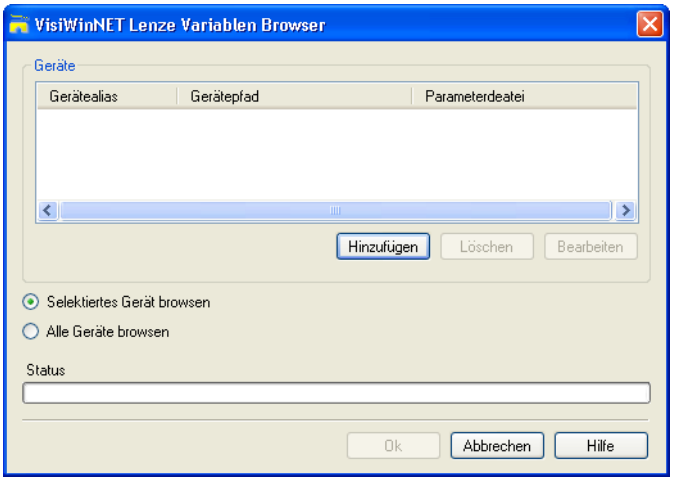
So gehen Sie vor:

1. Erstellen Sie im Projektexplorer einen neuen Kanal: Ordner **Variablen / Kanäle** markieren, mit der rechten Maustaste klicken und **Neu** wählen.
2. Klicken Sie in der Zeile **OPC-Server/Treiber** auf **[...]** und wählen Sie den Kommunikationstreiber **LenzeCAN**.
3. Schließen Sie den Dialog mit **OK**.
   Der Kanal wird angelegt und im Projektbaum angezeigt.
4. Klicken Sie im Objektbaum doppelt auf den neuen Kanal und konfigurieren Sie ihn auf der Eigenschaftenseite.



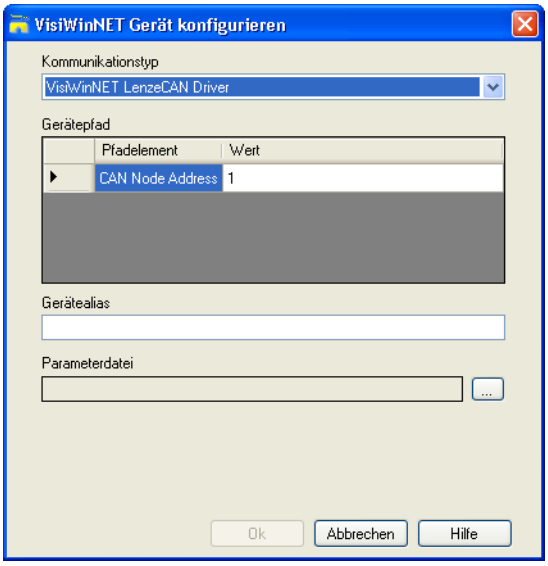
Variablen browsen

5. Markieren Sie im Projektexplorer den **CAN-Kanal**, klicken mit der rechten Maustaste und wählen **Variablen browsen**.



VWNS-035

6. Klicken Sie auf **Hinzufügen** und füllen Sie den Dialog "Gerät konfigurieren" aus.



VWNS-036

Kommunikationstyp: Kommunikationstyp "VisiWinNET LenzeCAN Driver"

Gerätepfad: CAN-Knotenadresse des Feldbus-Teilnehmers, mit dem Daten ausgetauscht werden sollen.

Parameterdatei: EDS-/GDC-Datei zum Import von Variablen

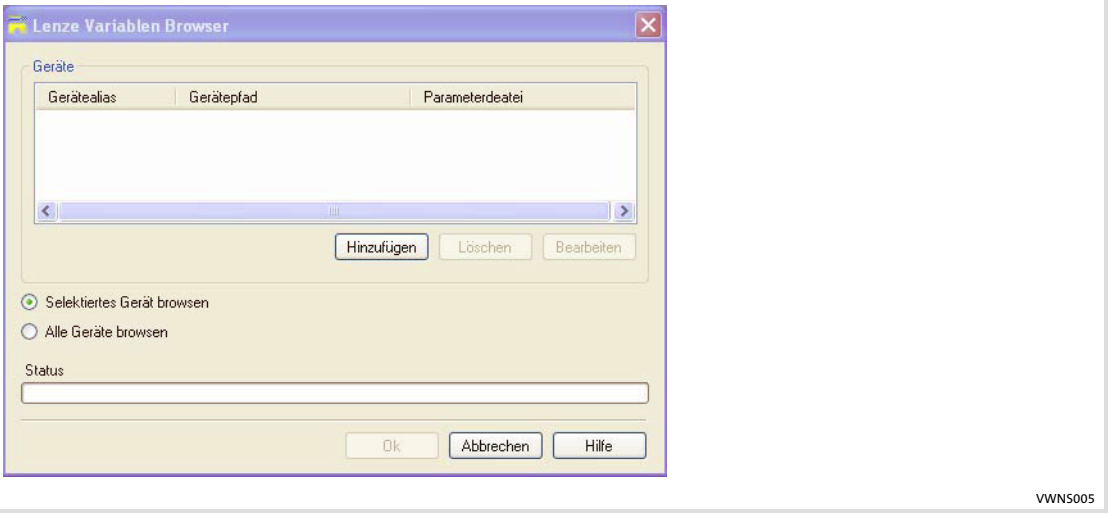
Für den Import von Codestellen über eine GDC-Datei ist immer eine pdb-Datei erforderlich, die dem Gerät entspricht, für das die GDC-Datei erzeugt wurde. pdb-Dateien liegen im GDC-Hauptverzeichnis in den Ordnern pdb/049 (Deutsch) bzw. pdb/044 (Englisch).

Gerätealias: Beliebigen Name eingeben.

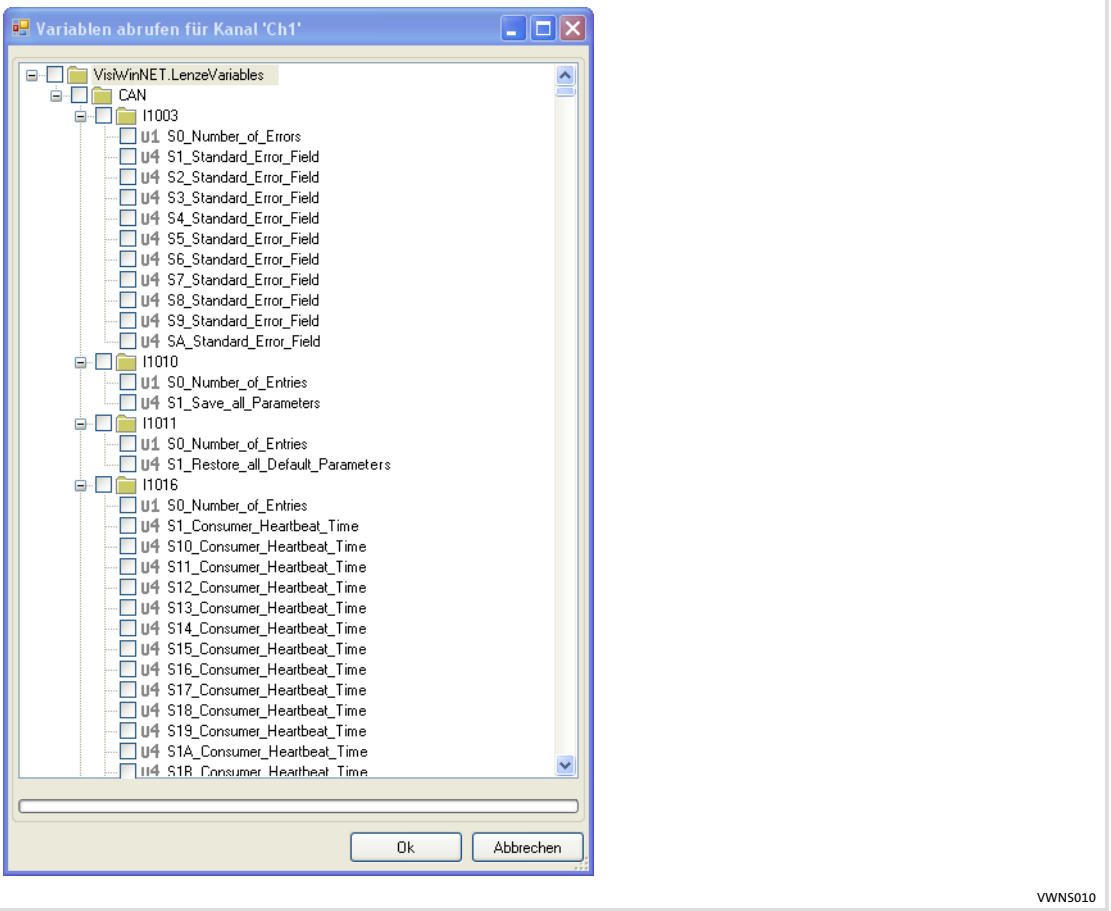
7. Schließen Sie den Dialog mit **OK**.

Variablen einlesen

8. Markieren Sie das Gerät, von dem die Variablen eingelesen werden soll.

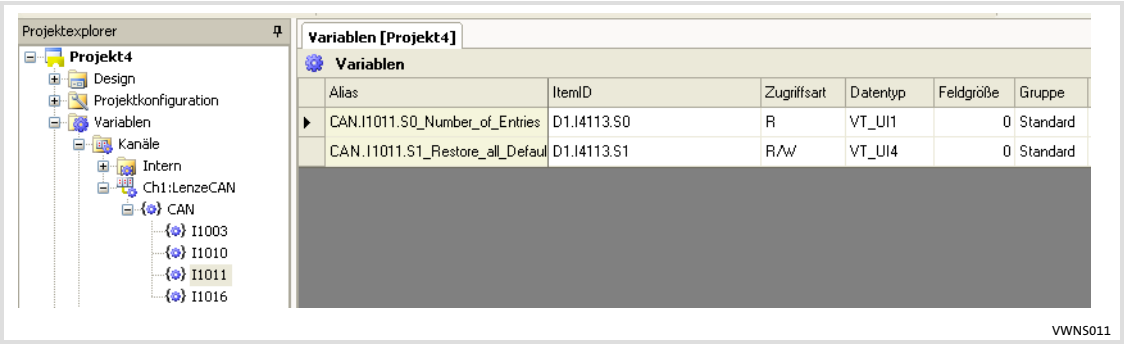


9. Schließen Sie den Dialog mit **OK**.
10. Markieren Sie im folgenden Dialog die zu importierenden Variablen.



11. Schließen Sie den Dialog mit **OK**.

Die ausgewählten Variablen stehen jetzt im VisiWinNET® Smart zur Verfügung.



5.2.3 CAN-Verbindung konfigurieren über PDOs

Neben dem Zugiff auf Codestellen können auch CAN-Prozessdaten (PDOs) vom Treiber verwaltet werden. Insgesamt können 10 PDOs wahlweise mit Rx- bzw. Tx-Funktion eingestellt werden. Als Übertragunsarten stehen ”Sync”, ”Ereignisgesteuert” und ”Zyklisch” zur Verfügung. (Weitere Informationen siehe Lenze Kommunikationshandbuch.)

Im folgenden Beispiel zeigen wir, wie ein PDO konfiguriert wird.

Aufgabe: Ein Byte-Wert soll zyklisch (Zeitintervall 1 s) vom EL1xx mit dem Identifier ”700dez” auf dem CAN-Bus ausgegeben werden. Ein Tx-PDO wird immer mit einer Länge von acht Byte ausgegeben. In diesem Beispiel soll das Byte im Byte 1 des Telegramms ausgegeben werden.

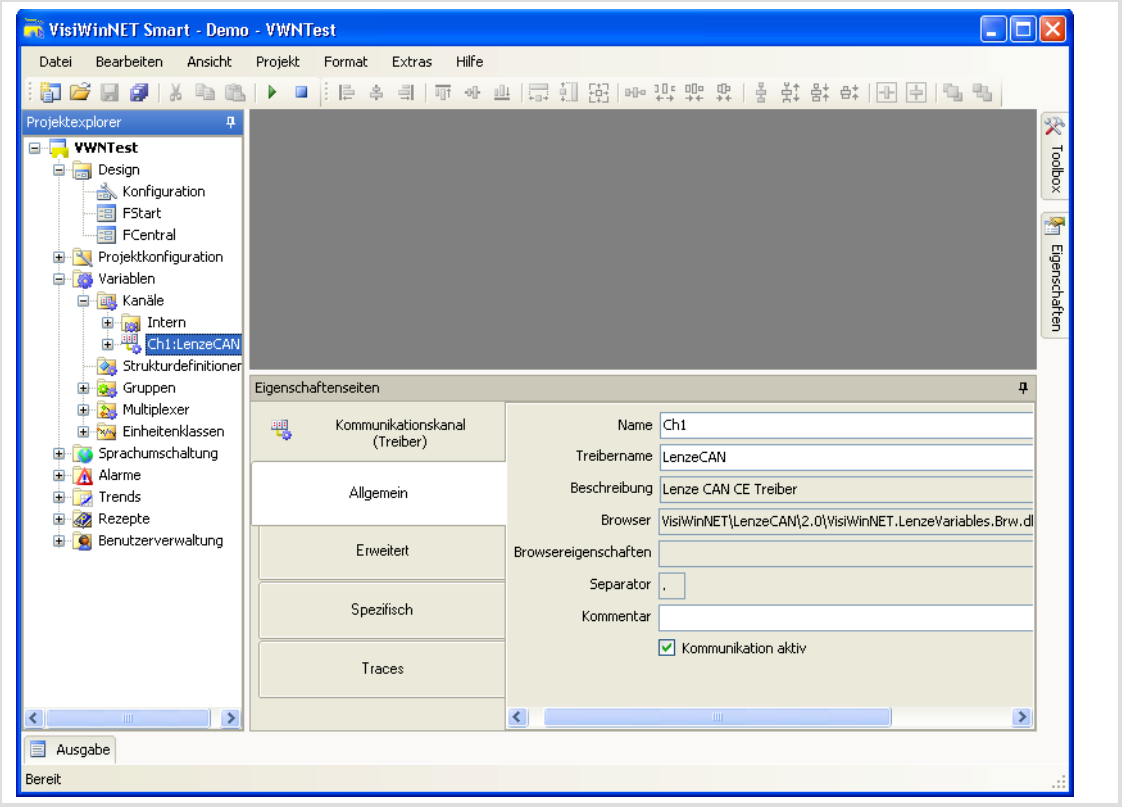
Kanal konfigurieren

Voraussetzung:

- ► Das EL1xx ist über LAN mit dem PC, auf dem VisiWinNET® Smart installiert ist, verbunden.
- ► Das EL1xx ist über CAN mit dem Antriebsregler verbunden.
  Entsprechende Informationen entnehmen Sie der Dokumentation zum EL1xx bzw. zum Antriebsregler.

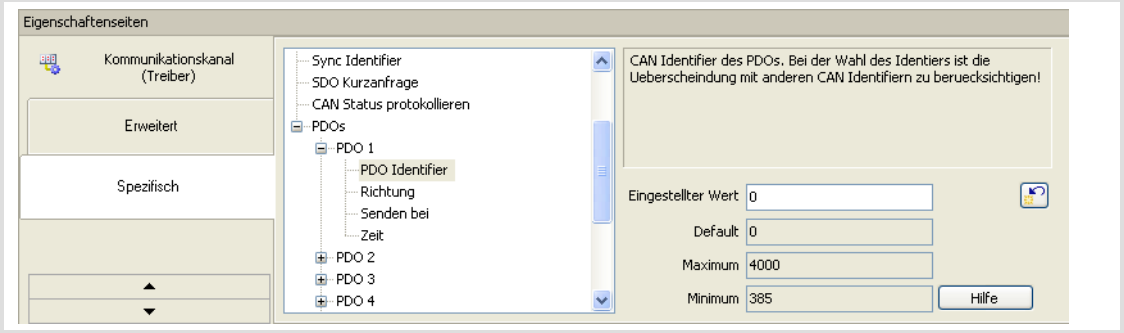
So gehen Sie vor:

1. Erstellen Sie im Projektexplorer einen neuen Kanal: Ordner **Variablen / Kanäle** markieren, mit der rechten Maustaste klicken und **Neu** wählen.
2. Klicken Sie in der Zeile **OPC-Server/Treiber** auf **[...]** und wählen Sie den Kommunikationstreiber **LenzeCAN**.
3. Schließen Sie den Dialog mit **OK**.
   Der Kanal wird angelegt und im Projektbaum angezeigt.
4. Klicken Sie im Objektbaum doppelt auf den neuen Kanal und konfigurieren Sie ihn auf der Eigenschaftenseite.



PDO-spezifische Einstellungen vornehmen

5. Aktivieren Sie das Register **Spezifisch** und öffnen Sie dort den Knoten **PDOs/PDO/PDO Identifier**.



Geben Sie dort folgende Werte ein:

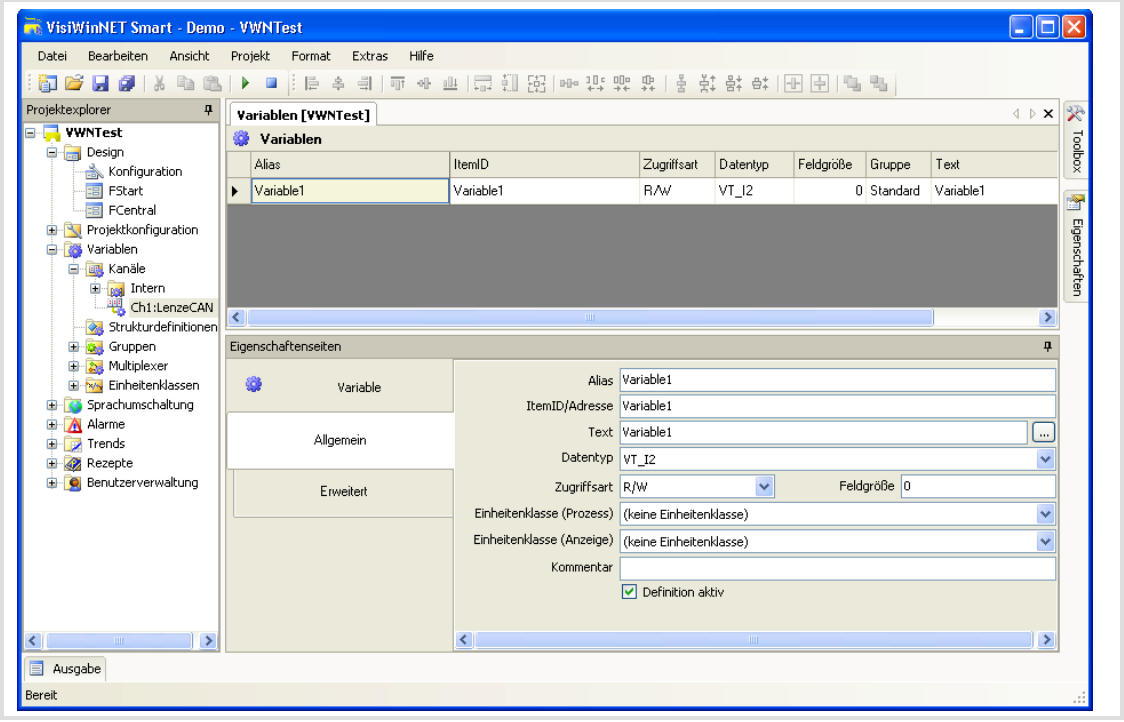
PDO Identifier: 700_{dez}
Richtung: Tx
Senden bei: Timer
Zeit: 1000ms

Alle notwendigen Einstellungen bzgl. des PDO-Verhaltens sind nun definiert. Nach den PDO-spezifischen Einstellungen ist es erforderlich, das PDO als eine VisiWinNET-Variable zur Anzeige bzw. Eingabe von Werten anzulegen.

PDO-Variable anlegen

6. Klicken Sie auf dem Clientbereich mit der rechten Maustaste und wählen Sie den Kontext-Menüpunkt **Neu**.

   Eine Variable wird angelegt.



7. Konfigurieren Sie die Variable wie folgt:

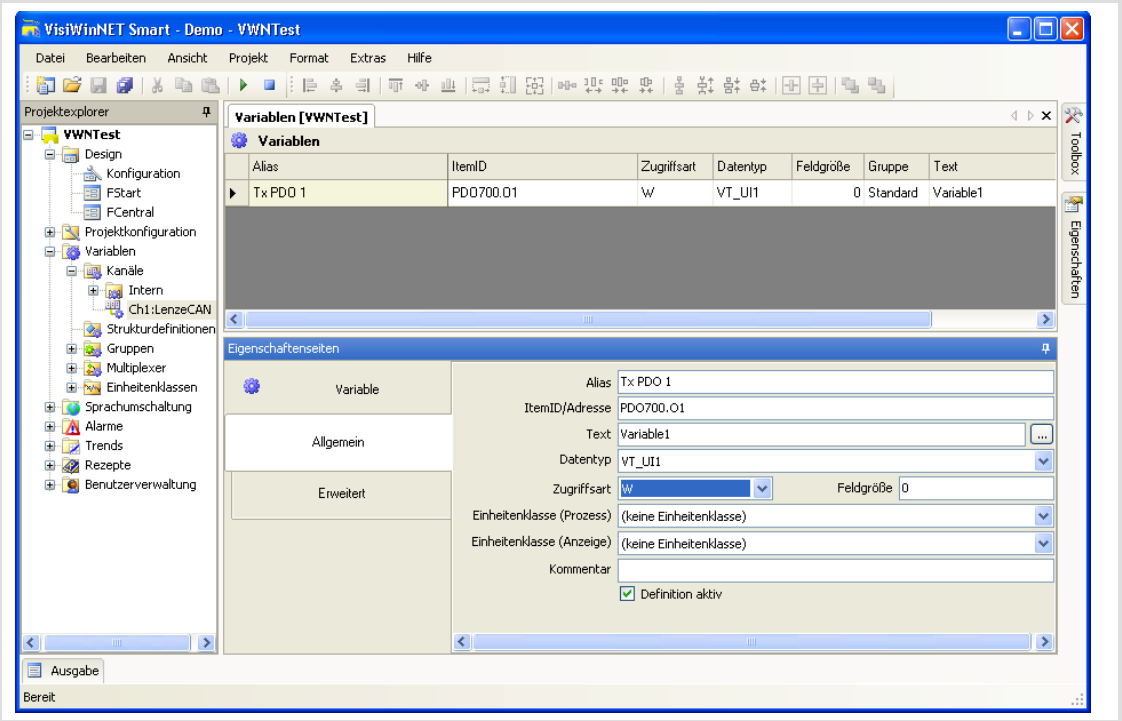
   Alias: Unter dem Alias wird die Variable in VisiWinNET zur Weiterverarbeitung zur Verfügung gestellt. In diesem Beispiel wird als Alias der Name ”Tx PDO 1” vergeben.

   ItemID/Adresse: Über die Item-ID wird die Verbindung zum PDO hergestellt. Der Eintrag erfolgt nach folgender Syntax: PDOxxx.Oy.

   Für xxx wird der PDO Identifier eingetragen. Das y steht für einen Offset im PDO-Telegramm. Das heißt, bei einem Offset von 0 erfolgt die Variablenabbildung ab dem Byte ”0”. Bezogen auf dieses Beispiel, bei dem der Identifier mit 700 definiert ist und die Byte Variable im Byte 1 abgebildet werden soll, ergibt sich folgender Eintrag für die ItemID/ Adresse: PDO700.O1

   Datentyp: Festlegung des Datentyps; hier ”VT_UI1”

   Zugriffsart: Festlegung der Zugriffsart; hier ”W” (Write)



Nun kann die Variable Tx PDO 1 als Item (z. B. als Eingabefeld ”VarIn”) weiterverwendet werden.

5.2.4 MPI-Verbindung konfigurieren

Über die MPI-Schnittstelle (multi point interface) kann ein ”EL1xx MPI” mit einer S7-SPS kommunizieren.

Kanal konfigurieren

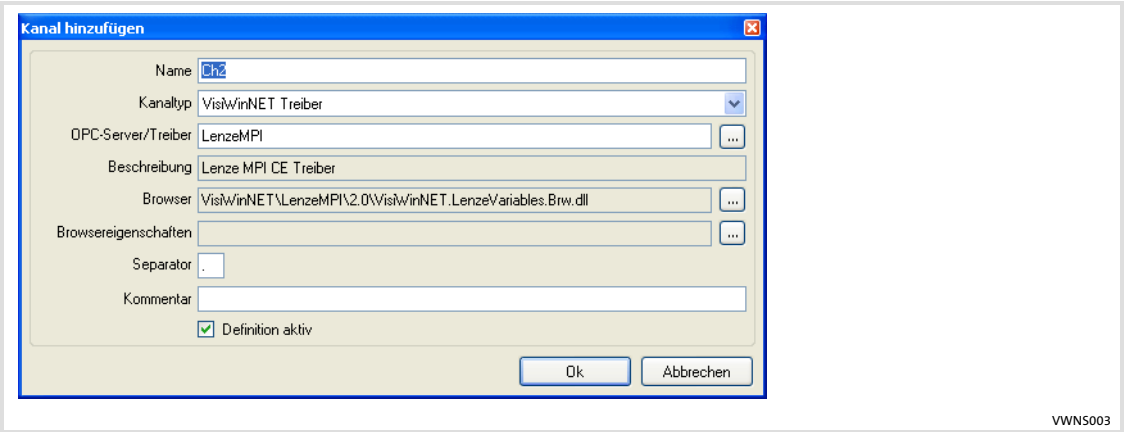
Voraussetzung:

- ► Das EL1xx ist über LAN mit dem PC, auf dem VisiWinNET® Smart installiert ist, verbunden.
- ► Das EL1xx ist über MPI mit der SPS verbunden.
  Entsprechende Informationen entnehmen Sie der Dokumentation zum EL1xx bzw. zur SPS.

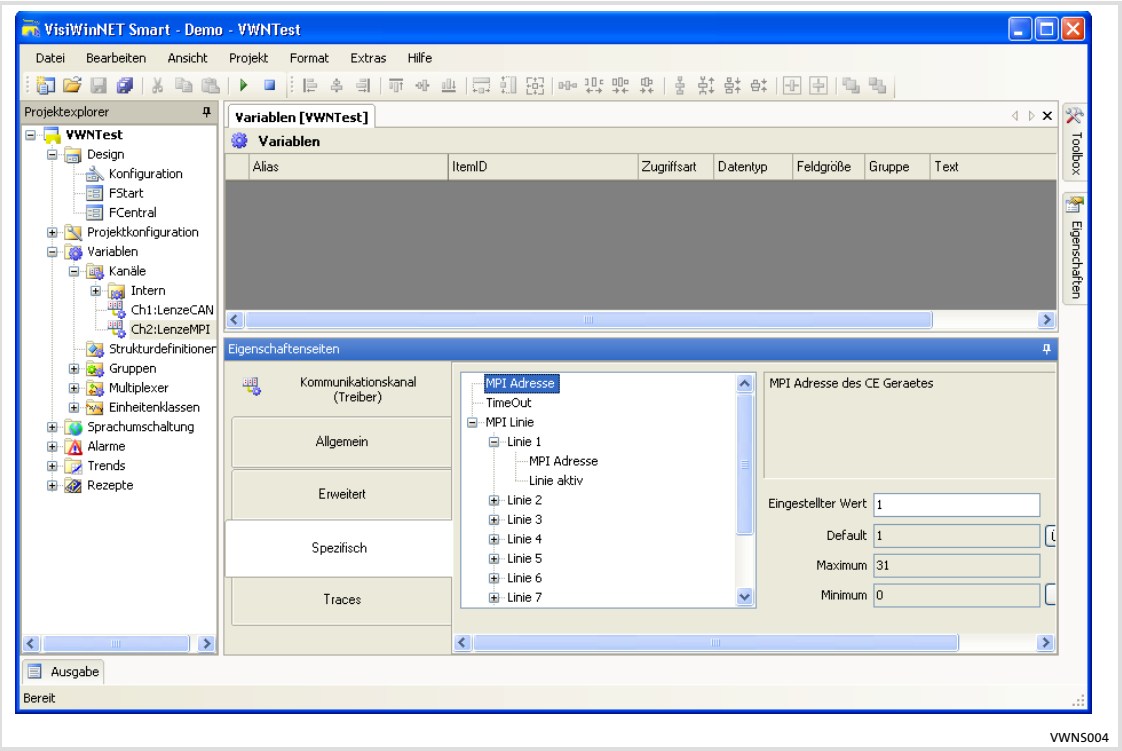
So gehen Sie vor:

Voraussetzung:

1. Erstellen Sie im Projektexplorer einen neuen Kanal: Ordner **Kanäle** markieren, mit der rechten Maustaste klicken und **Neu** wählen.
2. Klicken Sie In der Zeile **OPC-Server/Treiber** auf **[...]** und wählen Sie den Kommunikationstreiber **LenzeMPI**.



3. Schließen Sie den Dialog mit **OK**.
   Der Kanal wird angelegt und im Projektbaum angezeigt.
4. Klicken Sie doppelt auf den neuen Kanal im Objektbaum und konfigurieren Sie ihn auf der Eigenschaftenseite.



VWNS004

MPI Adresse: Hier ist die Adresse des EL1xx einzustellen. Diese Adresse muss mit der im MPI-Applet auf dem EL1xx eingestellten Geräteadresse übereinstimmen.

TimeOut: Einstellmöglichkeit bzgl. Timeout-Verhalten. Default ist eine Sekunde.

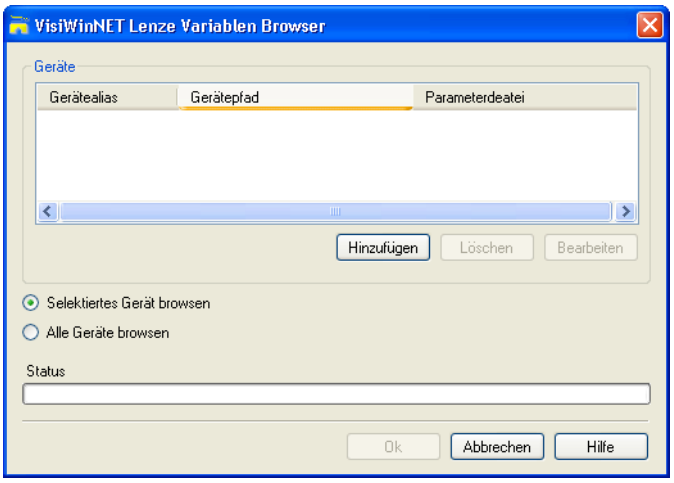
MPI Linie: Als MPI-Linie wird eine Kommunikationsbeziehung zwischen dem EL1xx und einer SPS bezeichnet. Maximal 10 Linien können aufgebaut werden; das heißt, ein EL1xx kann mit max. 10 SPSen Daten per MPI austauschen.

MPI Adresse: MPI-Adresse der SPS.

Linie aktiv: Aktiviert bzw. deaktiviert eine MPI-Linie. Durch Vorgabe von ”1” im Feld ”Eingestellter Wert” wird die Linie aktiviert.

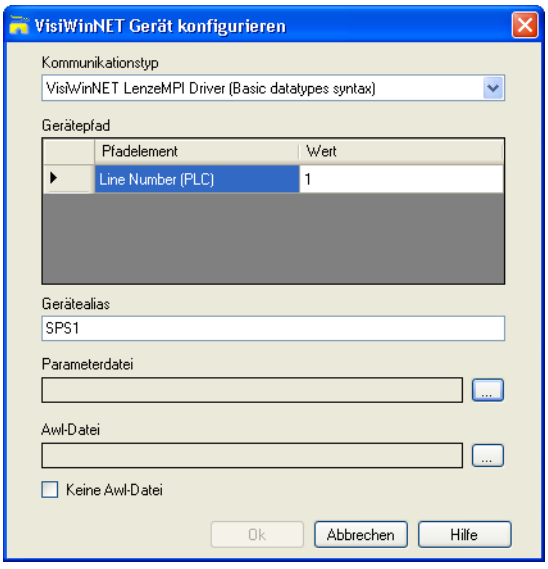
Variablen browsen

5. Markieren Sie im Projektexplorer den **MPI-Kanal**, klicken mit der rechten Maustaste und wählen **Variablen browsen**.



VWNS005

6. Klicken Sie auf **Hinzufügen** und füllen Sie den Dialog ”Gerät konfigurieren” aus.



VWNS006

Kommunikationstyp: Kommunikationstyp ”VisiWinNET Driver (Basic datatypes syntax)“ oder ”VisiWinNET LenzeMPI Driver (Instruction list /AWL syntax)“.

Gerätealias: Beliebiger Name

Parameterdatei: sdf-Datei zum Import von Variablen aus einer S7-Symboldatei (📖 112).

Awl-Datei: AWL-Datei, die eingelesen werden soll (📖 113).

keine AWL-Datei: Das Kontrollfeld muss markiert werden, wenn keine AWL-Datei eingelesen werden soll. Andernfalls kann der Dialog nicht beendet werden, ohne dass eine AWL-Datei zugewiesen wurde.

Zum Browsen der Variablen sind sdf-Dateien (generiert aus Symboltabelle) bzw. awl-Dateien (generiert aus Datenbaustein) erforderlich, die aus Step7 erzeugt werden können. Im Folgenden werden bereits generierte Dateien importiert.

Hinweis!

Die Auswahl von ”Basic datatypes syntax“ oder ”Instruction list /AWL syntax” hat ausschließlich Auswirkungen auf die Darstellung der Item-ID von Datenbausteinvariablen.

Der Unterschied soll an folgendem Beispiel für einen Merker M0.0 dargestellt werden:

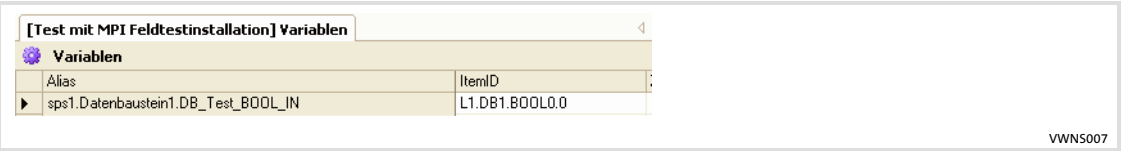


Abb. 5-1 Darstellung nach „Basic datatypes syntax“

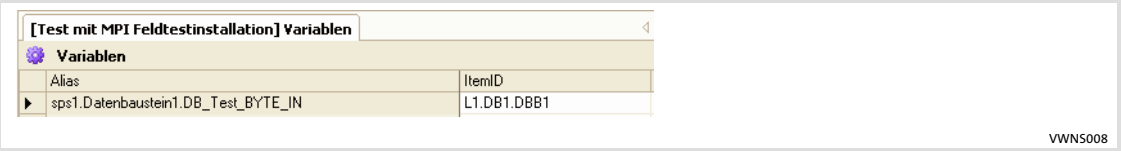
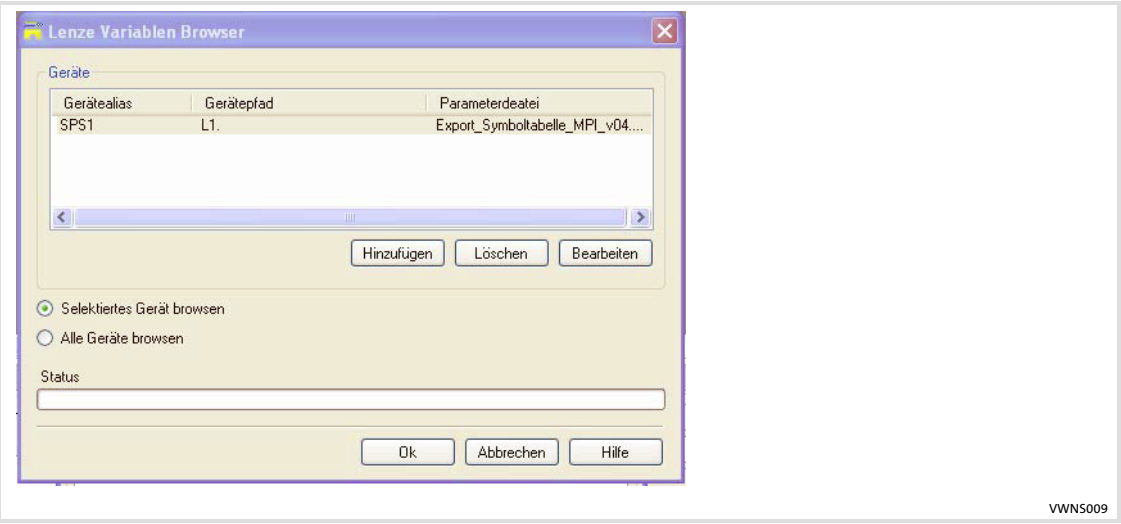


Abb. 5-2 Darstellung nach „Instruction list /AWL syntax“

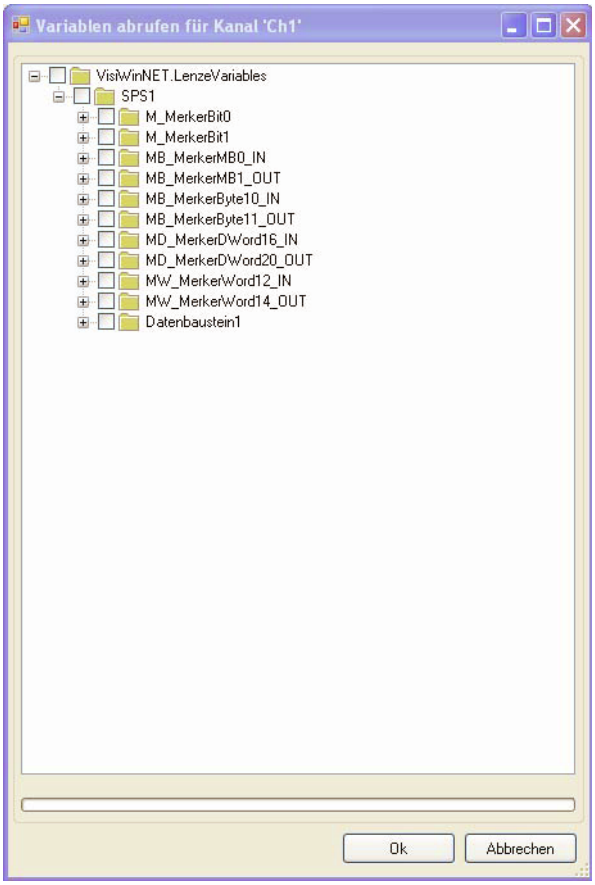
7. Schließen Sie den Dialog mit **OK**.

Variablen einlesen

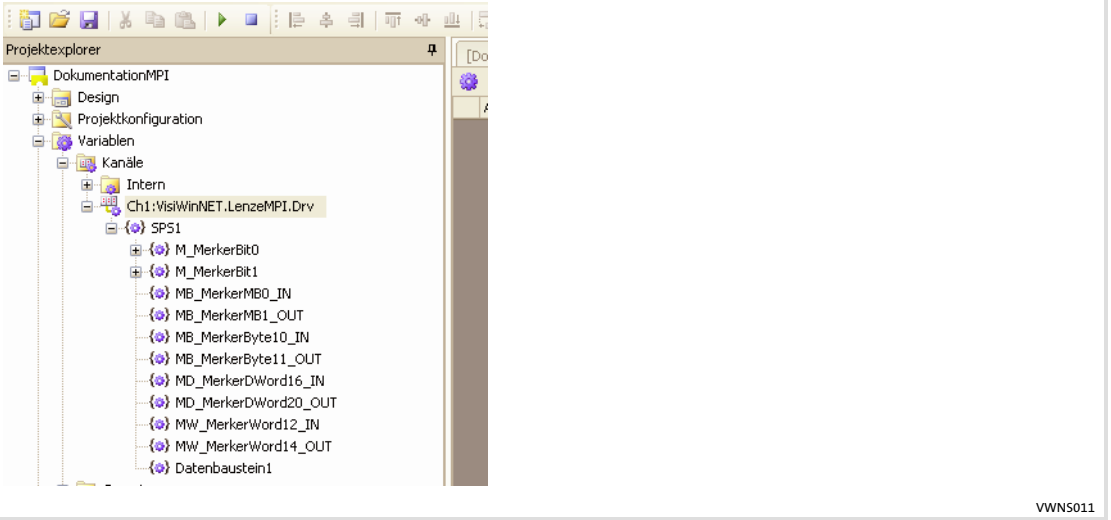
8. Markieren Sie das Gerät, von dem die Variablen eingelesen werden soll.



9. Schließen Sie den Dialog mit **OK**.
10. Markieren Sie im folgenden Dialog die zu importierenden Variablen.



11. Schließen Sie den Dialog mit **OK**.
    Die ausgewählten Variablen stehen jetzt im VisiWinNET® Smart zur Verfügung.



5.2.5 L-force Logic (CoDeSys)-Verbindung konfigurieren

Über VisiWinNET® Smart V 6.2 (oder höher) kann ein "EL1xx PLC" mit einer L-force Logic (CoDeSys)-Applikation kommunizieren.

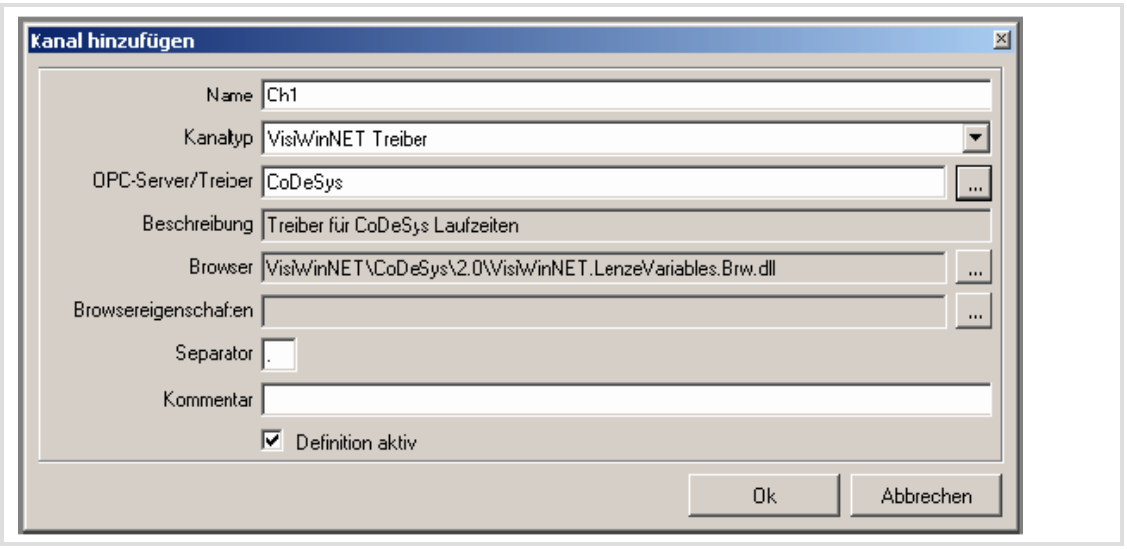
Voraussetzung:

► VisiWinNET® Smart und L-force Logic (CoDeSys) sind auf dem gleichen PC installiert oder aber auf unterschiedlichen PCs, die über LAN miteinander verbunden sind.

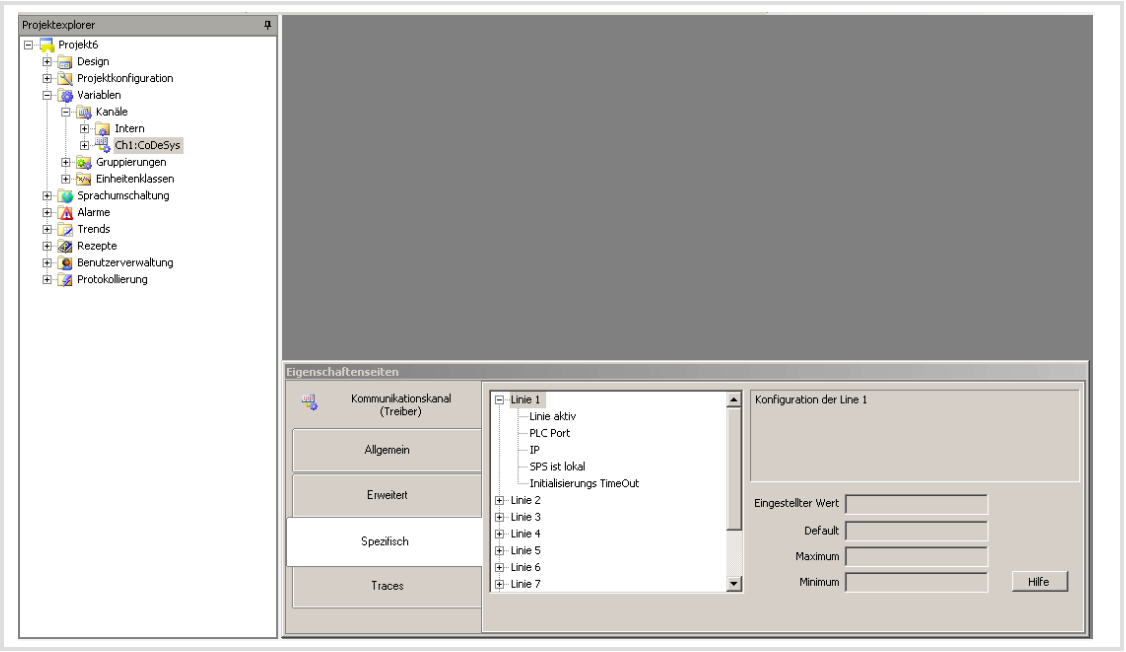
Kanal konfigurieren

So gehen Sie vor:

1. Erstellen Sie im Projektexplorer einen neuen Kanal: Ordner **Variablen / Kanäle** markieren, mit der rechten Maustaste klicken und **Neu** wählen.
2. Klicken Sie in der Zeile **OPC-Server/Treiber** auf **[...]** und wählen Sie **CoDeSys**.



3. Schließen Sie den Dialog mit **OK**.
   Der Kanal wird angelegt und im Projektbaum angezeigt.
4. Klicken Sie im Objektbaum doppelt auf den neuen Kanal und konfigurieren Sie ihn auf der Eigenschaftenseite.



Linie: Als Line wird ein Kommunikationsbeziehung zwischen dem EL1xx und einer L-force Logic (CoDeSys)-Applikation bezeichnet. Maximal 10 Linien können aufgebaut werden. Das heißt, ein EL1xx PLC kann mit max. 10 L-force Logic (CoDeSys)-Applikationen Daten austauschen.

Linie aktiv: Durch Vorgabe von TRUE im Feld „Eingestellter Wert“ wird die Linie aktiviert, durch FALSE deaktiviert.

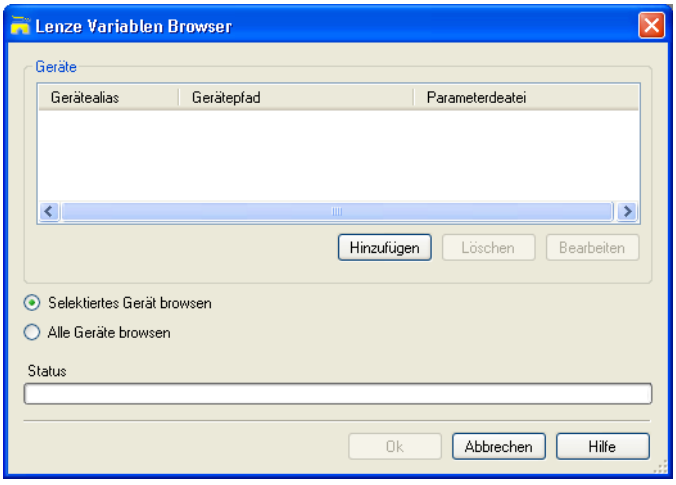
PLC Port: Von der SPS genutzter TCP/IP-Port.

IP: IP-Adresse des SPS-Laufzeitsystems. Die IP-Adresse des Laufzeitsystems entspricht der IP-Adresse des EL1xx PLC.

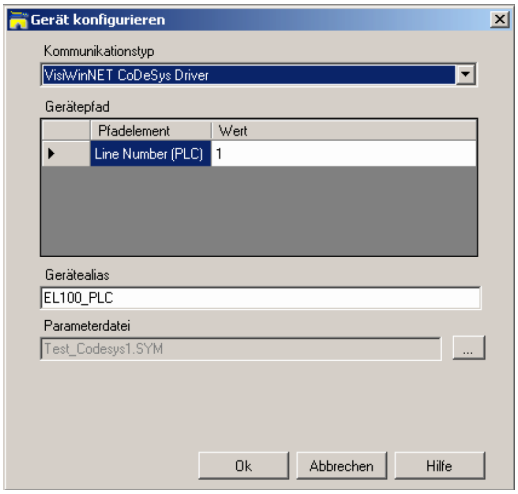
SPS ist lokal: Bei der Vorgabe TRUE greift die VisiWinNET®Smart-Applikation auf das lokale SPS-Laufzeitsystem zu. Die Einstellungen unter ”IP” werden ignoriert.

Variablen browsen

5. Markieren Sie im Projektexplorer den **CoDeSys-Kanal**, klicken mit der rechten Maustaste und wählen **Variablen browsen**.



6. Klicken Sie auf **Hinzufügen** und füllen Sie den Dialog "Gerät konfigurieren" aus.



Kommunikationstyp: Kommunikationstyp "VisiWinNET CoDeSys Driver"

Gerätepfad: Vorgabe der Linie, d. h. der CoDeSys-Applikation mit der die gebrowsten Variablen ausgetauscht werden sollen.

Gerätealias: Beliebigen Name eingeben.

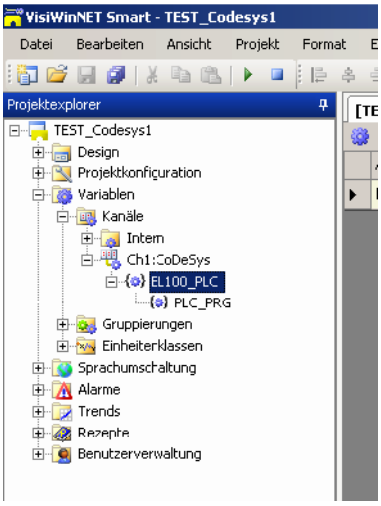
Parameterdatei: Name der sym-Datei , die vom L-force Logic (CoDeSys) erzeugt werden kann und für die Visualisierung des SPS-Programmes auf dem EL 1xx PLC benötigt wird (📖 116). Die sym-Datei zum PLC-Programm finden Sie im Projektordner L-force Logic (CoDeSys).

7. Schließen Sie den Dialog mit **OK**.

Variablen einlesen

8. Markieren Sie das Gerät, von dem die Variablen eingelesen werden soll.

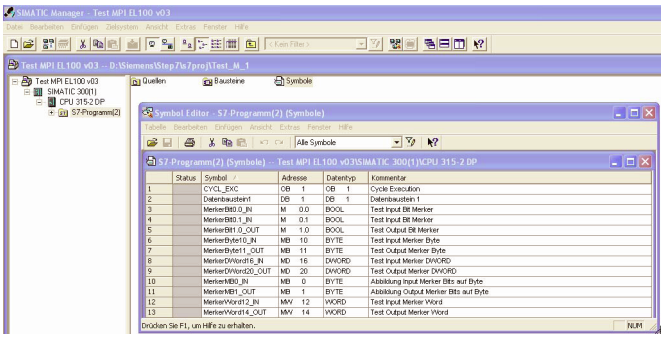
9. Schließen Sie den Dialog mit **OK**.
10. Markieren Sie im folgenden Dialog die zu importierenden Variablen.
11. Schließen Sie den Dialog mit **OK**.
    Die ausgewählten Variablen stehen jetzt im VisiWinNET® Smart zur Verfügung.



6 Anhang

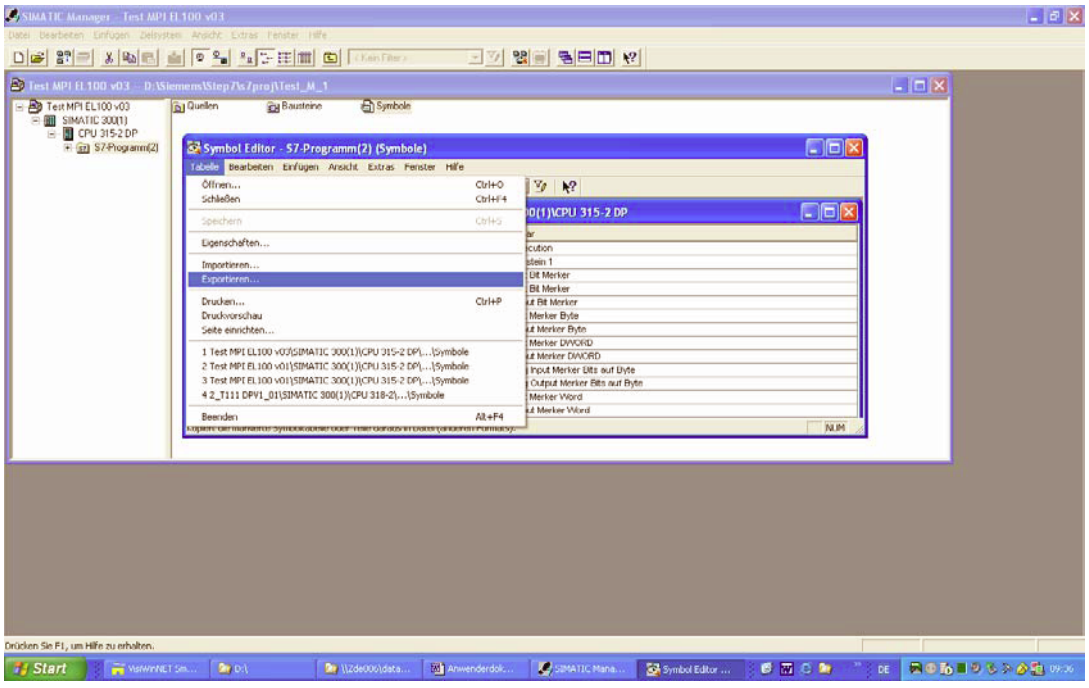
6.1 sdf-Datei aus einer Symboltabelle erzeugen

1. Öffnen Sie im ”Simatic-Manager”die Symboltabelle durch Doppelklick auf das Symbol ”Symbol”.



VWNS016

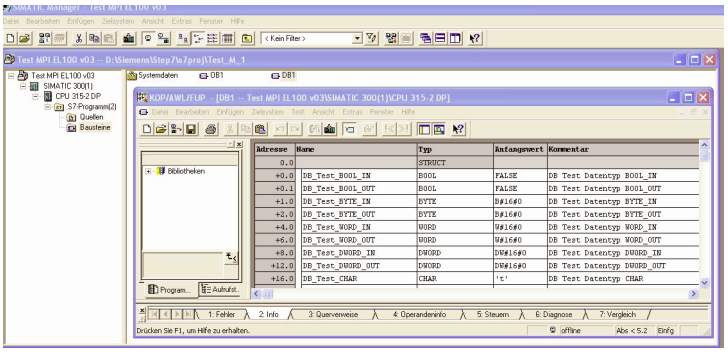
2. Öffnen Sie über **Tabelle** → **Exportieren** den Dialog ”Neu”, geben Sie im Feld ”Objektname” den Dateinamen ein und bestätigen Sie mit **OK**.



VWNS017

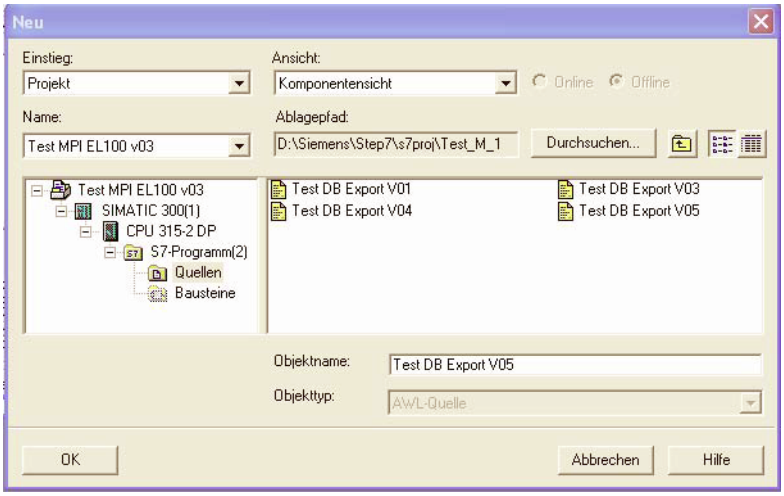
6.2 awl-Datei aus einem Datenbaustein erzeugen

1. Im ”Simatic-Manager” den Datenbaustein durch einen Doppelklick auf das DB-Symbol öffnen.



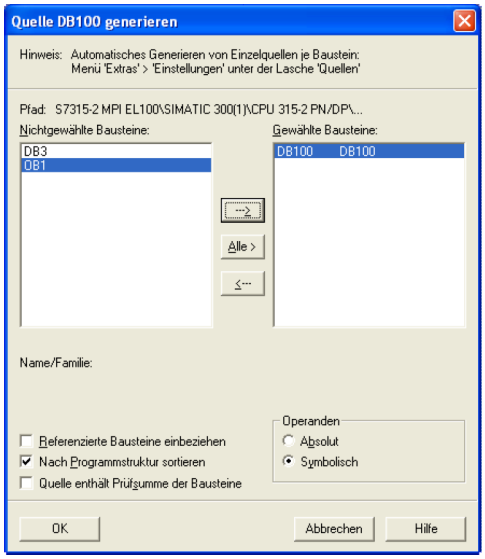
VWNS012

2. Über **Datei** → **Quelle generieren** den Dialog ”Neu” öffnen. Im Feld ”Objektname” den Dateinamen eingeben und mit **OK** bestätigen.

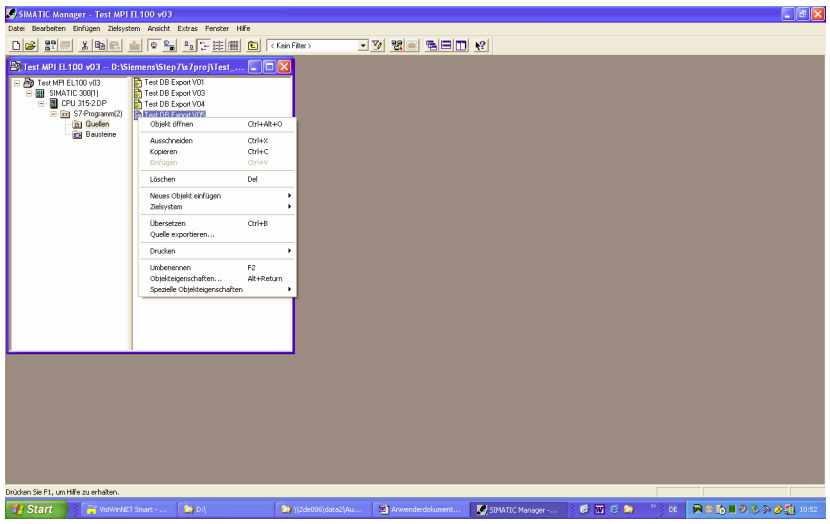


VWNS013

3. Bausteine wählen, die in der Quelle enthalten seien sollen und im Optionsfeld ”Operanden” den Eintrag ”Symbolisch” markieren.
   Diese Bausteine können anschließend in VisiWinNET verwendet werden.

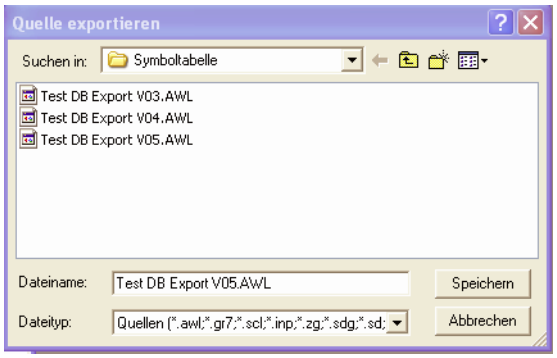


4. Im Projektbaum die neu generierte Quelle markieren, rechten Mausklick ausführen und **Quelle exportieren** aktivieren.



VWNS014

5. Einen Namen für die zu exportierende Quelle eingeben, unter ”Dateiname” und als ”Dateityp” ”.awl” auswählen und mit **Speichern** abschließen.



VWNS015

Die gespeicherte awl-Datei kann nun im VisiWinNET eingelesen werden.

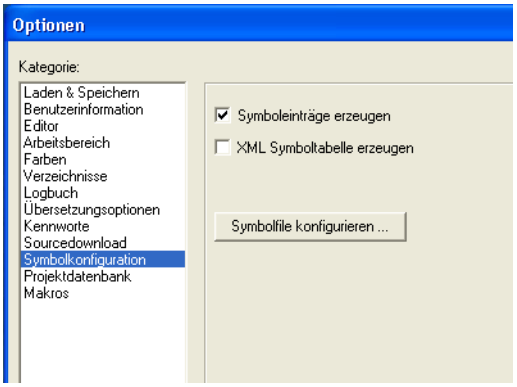
Die Symbolaktivierung erfolgt in der Entwicklungumgebung des SimaticManagers.

6.3 sym-Datei im PLC-Designer erzeugen

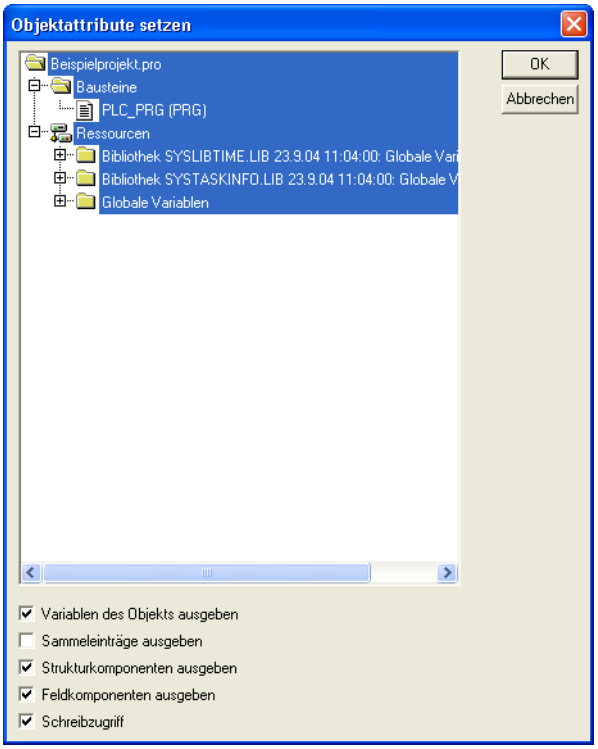
Eine sym-Datei benötigen Sie, wenn die Visualisierung, z. B. VisiWinNet® Smart, auf das SPS-Programm zugreifen soll. Eine sym-Datei beinhaltet alle Variablen und Deklarationen des SPS-Programms.

So gehen Sie vor:

1. Öffnen Sie im PLC-Designer das Menü **Projekt** und wählen Sie dort den Menüpunkt **Optionen**.
   Der Dialog ”Optionen” wird geöffnet.



2. Wählen Sie im Dialog ”Optionen” die Kategorie ”Symbolkonfiguration”.
3. Markieren Sie dort das Kontrollfeld **Symboleinträge erzeugen** und klicken Sie anschließend auf die Schaltfläche **Symbolfile konfigurieren**.
   Der Dialog ”Objektattribute setzen” wird geöffnet.



4. Markieren Sie im Dialog ”Objektattribute setzen” unbedingt das Kontrollfeld *Variablen des Objekts ausgeben*. Belassen Sie alle anderen Kontrollfelder wie vorgegeben.
5. Markieren Sie im Objektbaum alle Objekte, die Variablen für die Visualisierung beinhalten und mit Variablen des Objekts aktiviert werden.

   Exportieren Sie nur die Variablen, die für die Visualisierung benötigt werden. Je kleiner die sym-Datei ist, desto kürzer ist die Übertragungszeit auf das Zielsystem.
6. Schließen Sie den Dialog ”Objektattribute setzen” und den Dialog ”Optionen” jeweils mit **OK**.
7. Öffnen Sie in der Menüleiste das Menü **Projekt** und wählen Sie dort den Menüpunkt **Alles bereinigen**.
8. Öffnen Sie das Menü **Projekt** erneut, wählen Sie jetzt aber den Menüpunkt **Alles übersetzen**.

   Das Projekt wird überprüft, übersetzt und unter dem Dateinamen ”Beispielprojekt.sym” am vorgegebenen Speicherort abgelegt.

6.4 Skripte

6.4.1 Grundlagen

VisiWinNET® Smart unterstützt die Erstellung individueller Programmlogik über die Programmierung von benutzerdefinierten Funktionen mit Hilfe der Programmiersprache VB.NET. Die benötigten Funktionen, die im Allgemeinen auch als Skripte bezeichnet werden, können über einen integrierten Codeeditor in unterschiedlichen Codemodulen hinterlegt werden.

Im Gegensatz zu einem herkömmlichen Skript werden die mit VisiWinNET® Smart erstellten Funktionen allerdings nicht zur Laufzeit des Programms interpretiert, sondern bereits während der Projekterstellung in ausführbaren .NET Code übersetzt. Dadurch ergibt sich eine höhere Ausführungsgeschwindigkeit gegenüber klassischen Skripten und darüber hinaus der Vorteil, dass der gesamte Funktionsumfang des .NET Frameworks innerhalb der Codemodule genutzt werden kann.

Ein Codemodul kann entweder global hinterlegt werden, oder einem Formular zugeordnet sein.

Global hinterlegtes Codemodul

In globalen Codemodulen werden Funktionen oder globale Variablen hinterlegt, die an verschiedenen Stellen im Programm benötigt werden.

Das Hinzufügen eines globalen Codemoduls wird über das Kontextmenü des Projektexplorerknotens **Design** angeboten:

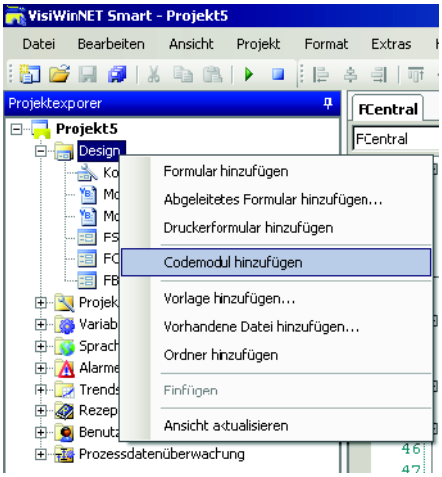


Abb. 6-1 Codemodul hinzufügen

Formular zugeordnetes Codemodul

Codemodule, die einem Formular zugeordnet sind, enthalten Funktionen, die innerhalb eines bestimmten Formulars benötigt werden. Die Ereignisse des Formulars und der darauf platzierten Steuerelemente und Komponenten können dabei innerhalb des Codemoduls abgefangen werden.

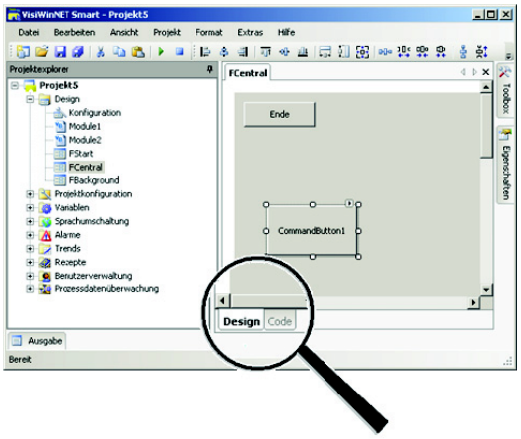


Abb. 6-2 Umschalten schwischen Formular-Ansicht (Design) und Code-Ansicht (Code)

Aufbau von Skripten

Ereignisse werden dazu verwendet, eine Funktion aufzurufen. Ein Ereignis wird im Codemodul durch eine Rahmenfunktion repräsentiert, in der ausführbarer Code geschrieben wird. Das Einfügen einer Rahmenfunktion wird durch die beiden Auswahllisten am oberen Rand des Codemoduls unterstützt.

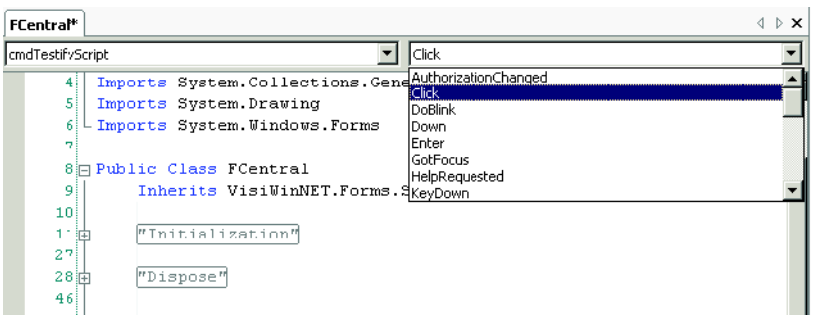


Abb. 6-3 Beispiel Codemodul

Die linke Auswahlliste beinhaltet die Namen aller Objekte, die Ereignisse auslösen können, also die Namen des Formulars und der Steuerelemente und Komponenten darauf. Nach der Auswahl eines Namens werden in der rechten Auswahlliste alle Ereignisse aufgeführt, die vom links ausgewählten Objekt unterstützt werden.

Beispiel: Das links ausgewählte CommandButton-Steuerelement mit dem Namen ”cmdTestifyScript” stellt unter anderem auch ein ”Click”-Ereignis zur Verfügung, das ausgelöst wird, wenn der Benutzer mit der linken Maustaste auf das Steuerelement klickt.

Rahmenfunktion einfügen

Durch die Auswahl eines Ereignisnamens rechts wird eine Rahmenfunktion eingefügt.

Beispiel für eine Ereignis-Rahmenfunktion:

```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles VarIn1.Change

End Sub
```

Das gewählte Objekt "VarIn1" stellt das "Change"-Ereignis zur Verfügung, das die Änderung des Variablenwertes signalisiert. Zum Einfügen der Funktion wurde in der linken Auswahlliste das Objekt "VarIn1" ausgewählt, anschließend in der rechten Auswahlliste "Change".

Der Funktionskopf der eingefügten Rahmenfunktion beinhaltet:

- Einen Namen für die Funktion ("VarIn1_Change"). Der Name muss innerhalb eines Codemoduls eindeutig sein.
- Die Parameter "sender" und "e", "sender" ist dabei üblicherweise ein Verweis auf das Objekt, in dem das Ereignis ausgelöst wurde. "e" beinhaltet üblicherweise weitere Informationen zum Ereignis.
- Die "Handles"-Klausel bezeichnet die Objekte und Ereignisse, durch die die Funktion angesprungen wird. In diesem Fall also nur das Change-Ereignis von VarIn1. Sie können aber weitere Ereignisse mit gleichen Parametern auf die Funktion umlenken, indem Sie, durch Kommata getrennt, weitere Ereignisbezeichener (<Objektname>.<Ereignisname>) an die "Handles"-Klausel anfügen. In diesem Fall wird klar, warum der "Sender"-Parameter mit in die Funktion übergeben wird. Über "Sender" kann dann das Objekt ermittelt werden, das dieses Ereignis ausgelöst hat.

Das Funktionsende wird durch "End Sub" repräsentiert. Zwischen Funktionskopf und Funktionsende ist der ausführbare VB-Code erlaubt:

```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles VarIn1.Change

    If e.Value > 50 Then
      VarIn1.DataBackColor = System.Drawing.Color.Red
    Else
      VarIn1.DataBackColor = System.Drawing.SystemColors.Window
    End If

End Sub
```

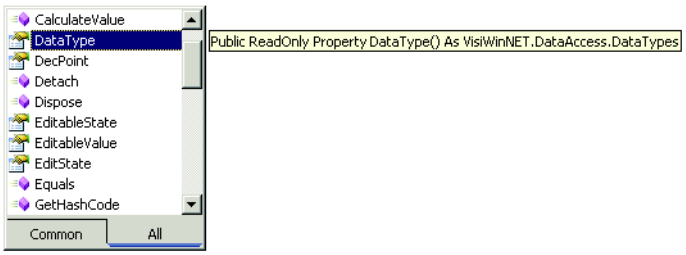
Beispiel: Innerhalb des e-Arguments werden weitere Informationen zum Ereignis zur Verfügung gestellt. Hier wird anhand der Value-Eigenschaft aus dem Variablen-Kern die Hintergrundfarbe im Datenbereich von "VarIn1" umgeschaltet.

Hilfe beim Erstellen von Skripten

VisiWinNET® Smart beinhaltet Programmierschnittstellen für die Oberflächenelemente und das darunterliegende System.

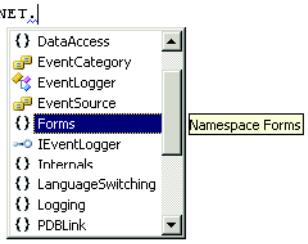
Welche Funktionen (Eigenschaften/Methoden/Ereignisse) ein Objekt zur Verfügung stellt, erfahren Sie in Kurzform über das im Quellcodeeditor implementierte Intellisense-Fenster, das sich automatisch bei der Eingabe von z. B. Objektnamen oder Namensräumen öffnet (Groß-/Kleinschreibung ist von Bedeutung!).

```
Private Sub VarIn2_VerifyEditValue(ByVal sender As Object, _
           ByVal e As VisiWinNET.DataAccess.VerifyEditValueEventArgs) Handles VarIn2.VerifyEditValue

    VarIn2.VWItem.
End Sub
```



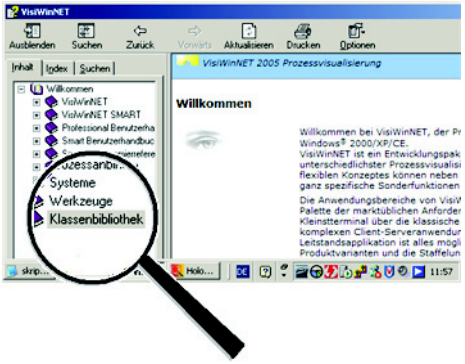
Über die Angabe des Objektnamens ”VarIn2” mit abschließendem ”.” werden alle Eigenschaften aufgelistet. Eigenschaften sind Objekte und beinhalten wiederum Eigenschaften, die über die Cursortasten ausgewählt und über Enter eingefügt werden.

```
Private Sub VarIn2_VerifyEditValue(ByVal sender As Object, _
           ByVal e As VisiWinNET.DataAccess.VerifyEditValueEventArgs) Handles VarIn2.VerifyEditValue

    VisiWinNET.
End Sub
```



Über die Angabe des Wortes ”VisiWinNET” werden Sie in die Namensräume geführt, die alle Visualisierungsfunktionenen thematisch sortiert enthalten.

Zusätzlich haben Sie für die Recherche in der Onlinehilfe eine Programmierreferenz, die die Schnittstellen des Produktes aus Sicht des Programmierers beschreibt.



Tipp!

Einen empfehlenswerten Einstieg in die Programmierung mit VB finden Sie unter ”http://openbook.galileocomputing.de/vb_net/”. Die passende Entwicklungsumgebung zum genannten Einstiegskurs wird unter der Bezeichnung ”Visual Basic Express Edition” von Microsoft kostenfrei angeboten.

Im folgenden Kapitel ”Beispiel-Funktionen” finden Sie außerdem einige ausgewählte Beispiele, die erfahrungsgemäß oft in Projekten verwendet werden.

Das .NET-Framework stellt eine Vielzahl von Funktionen zur Verfügung, die die gebräuchlichen Programmieraufgaben unter Windows unterstützen. Oftmals gelingt es über die Verwendung einer Suchmaschine ein Stückchen funktionierenden Quellcodes zu ergattern, das mit dem Framework arbeitet. Zusätzlich müssen Sie in den meisten Fällen aber auch noch eine Referenz auf die entsprechenden Assemblies im Projekt aufnehmen. Dies wird im Projektexplorer über den Knoten **Design** → **Referenzen** erledigt.

Bei der Recherche hilft das ”MSDN Online” (http://msdn.microsoft.com/de-de/default.aspx), das sowohl Informationen über die Programmiersprache als auch über das .NET Framework enthält.

6.4.2 Fehlersuche in Funktionen

Zwei Arten von Fehlern werden generell unterschieden:

- Syntaktische Fehler
- Laufzeitfehler

Syntaktische Fehler

Syntaktische Fehler oder fehlende Variablendeklarationen werden schon beim Erstellen des Projektes gemeldet. Das Erkennen und Beseitigen dieser Fehler ist für den Entwickler relativ unkompliziert, weil sie schon im Entwicklungsstadium gemeldet werden. Im Projekt enthaltene Programmierfehler werden beim Erstellen (über den Menüeintrag **Projekt → Erstellen**" oder den Shortcut **Strg+Umschalt+B**) im Ausgabefenster angezeigt. Enthält das Projekt Fehler, die vom Compiler erkannt werden, lässt es sich auch nicht starten.

Laufzeit-Fehler

Laufzeitfehler, wie "Division By Zero" oder fehlende Objektverweise werden erst dann erkannt, wenn die entsprechende Funktion ausgeführt wird. Hier liegt die Gefahr, dass der Code nach der Inbetriebnahme mit Daten aus dem Prozess gefüttert wird, die nicht getestet wurden und zu einer Ausnahme führen. Hier hilft die Verwendung der Try-Catch Klausel mit einem definierten Verhalten für den Fehlerfall.

```
Private Sub ItemServer1_Change(ByVal sender As System.Object, _
    ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles ItemServer1.Change
  Try
    Dim Divident As Double = _
      VisiWinNET.Services.AppService.VWGet("Ch1.myDivident")

    Dim Divisor As Double = e.Value

    Dim Quotient As Double = Divident / Divisor

    VisiWinNET.Services.AppService.VWSet("Ch1.myQuotient", Quotient)

  Catch ex As Exception 'ooh, an error
    If MsgBox("ItemServer1_Change results in: " _
          & vbCrLf & ex.Message _
          & vbCrLf & "Continue?", _
           MsgBoxStyle.YesNo) = MsgBoxResult.No Then

      'decision: End application
      VisiWinNET.Forms.ProjectForms.StartForm.ShutdownProject()

    End If
  End Try
End Sub
```

Beispiel: Tritt ein Laufzeitfehler bei der Berechnung "Divident/Divisor= Quotient" auf, springt der Programmlauf in die Catch-Klausel. Hier wird anhand einer Messagebox entschieden, ob die Applikation beendet werden soll.

6.4.3 Beispiel-Funktionen

Lesen und Schreiben von Variablenwerten

Ein typisches Szenario ist das Lesen von einer oder mehreren Variablen, wenn ein Trigger ausgelöst wird. Der Trigger ist dabei meist selbst eine Variable. Das Lesen/Schreiben wird über die Funktion VWGet/VWSet ausgeführt.

```
'Sample VWGet/VWSet: Reading a byte field from Demo OPC Server
'increment elements and write back
Private Sub ItemServerArrayRead_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs)
        Handles ItemServerArrayRead.Change
    If e.Value = 1 Then
      Dim xObj As Object = _
        VisiWinNET.Services.AppService.VWGet("Ch1.Simulation.Array.Byte")
      Dim x() As Byte = xObj
      For i As Integer = 0 To x.Length - 1
        Dim bt As Byte = x(i)
        bt = bt + 1
        x.SetValue(bt, i)
      Next
      VisiWinNET.Services.AppService.VWSet("Ch1.Simulation.Array.Byte", x)
    End If
  End Sub
```

Diese Funktion liest im Rahmen eines Change-Ereignisses von einer VarServer-Komponente das Feld "Simulation.Array.Byte" aus dem Demo-OPC-Server aus dem Variablenkern, inkrementiert die Elemente und schreibt das Datenfeld zurück. Weitere Beispiele zu den Funktionen "VWGet"/"VWSet" finden Sie in der Online-Hilfe.

Bits in ganzzahligen Variablenwerten auswerten

VisiWinNET® Smart beinhaltet Funktionen, die den VB-Programmierer beim Umgang mit Bits aus ganzzahligen Variablenwerten unterstützen.

```
'Toggles Bit 2 in process variable "MyBitModulatedValue"
If VisiWinNET.Services.AppService.VWGetBit("MyBitModulatedValue", 2) = True Then
    VisiWinNET.Services.AppService.VWSetBit("MyBitModulatedValue", 2, False)
Else
    VisiWinNET.Services.AppService.VWSetBit("MyBitModulatedValue", 2, True)
End If
```

Diese Funktion greift auf das Bit 2 in der Prozessvariable "MyBitModulatedValue" zu und ändert den Wert.

Ausgabeinformationen zum Testen

Um trotz des fehlenden Debuggers die Diagnose zu erleichtern, lenkt VisiWinNET® Smart Standardausgaben in das Ausgabefenster um. Standardausgaben werden über den Befehl Console.WriteLine zeilenweise ausgegeben.

```
Dim i As Integer = 21
Console.WriteLine(”Hello World, the answer is: ” & i * 2)
```

Die Ausgabe wird im Ausgabefenster vom VisiWinNET® Smart angezeigt.

Unter Windows CE muss folgender Code verwendet werden:

```
Private i As Integer = 0

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
   #if DEBUG
    Console.WriteLine(”TimerTick: ” & i)
     i = i + 1
   #End If
End Sub
```

Unter Design\Configuration\Build kann der Compilerschalter gesetzt werden.

Texte aus der Sprachumschaltung lesen

Über die Typen im Namensraum ”LanguageSwitching” sind die Informationen aus der Sprachumschaltung zu erreichen. Dazu gehört auch das Lesen einzelner Texte.

```
Dim s As String =_
      VisiWinNET.LanguageSwitching.Localization.GetText(”@FCentral.MyText”)
```

Im Editor der Sprachumschaltung wurde eine Textgruppe ”FCentral” angelegt und darin ein Text namens ”MyText”. Nach dem Funktionsaufruf steht der Text in der aktuell im Projekt eingestellten Sprache in der Variable ”s” zur Verfügung.

Weniger Code für viel Funktion (globale Module)

Das einführende Beispiel zur Umschaltung der Hintergrundfarbe eines ”VarIn”-Steuerelementes in Abhängigkeit vom Variablenwert im Kapitel ”Skripte” lässt sich unter Verwendung eines globalen Codemoduls auch auf mehrere Steuerelemente auf unterschiedlichen Formularen erweitern.

Fügen Sie dazu über das Kontexmenü des Projektexplorerknotens ”Design” ein neues Codemodul (”Module1”) in das Projekt ein. Fügen Sie in diesem Codemodul folgende Funktion ein:

```
Public Sub SetVarinDataBackColor(ByVal VarIn As VisiWinNET.Forms.VarIn, _
            ByVal e As VisiWinNET.DataAccess.ChangeEventArgs)
    If e.Value > 50 Then
      VarIn.DataBackColor = System.Drawing.Color.Red
  Else
    VarIn.DataBackColor = System.Drawing.SystemColors.Window
  End If
End Sub
```

Als Funktionsparameter werden das auslösende VarIn-Objekt und das im Change-Ereignis übergebene ”ChangeEventArgs” -Objekt erwartet.

Der Aufruf auf die Funktion aus einem Change-Ereignis reduziert sich damit auf eine einzige Zeile und sieht dann so aus:

```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
            ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) _
            Handles VarIn1.Change
  Module1.SetVarinDataBackColor (sender, e)
End Sub
```

Vorteile der Verwendung der globalen Funktion:

- Weniger Code beim Aufruf
- Bessere Parametrierung möglich (wenn z. B. der Grenzwert ”50” zentral geändert werden soll)

6.4.4 Einschränkungen in der Programmierschnittstelle von VisiWinNET Smart

Folgende Einschränkungen gibt es bei der Programmierschnittstelle von VisiWinNET® Smart:

- VisiWinNET® Smart beinhaltet keinen eigenen Debugger. Fehler werden entweder beim Erstellen angezeigt oder führen zur Laufzeit zu einer Ausnahme. Wenn in besonders hartnäckigen Fällen ein Debuggen unausweichlich erscheint, kann mithilfe des CLR-Debuggers (Bestandteil des Microsoft .NET Framework SDK 2.0) gearbeitet werden. Für Projekte unter Windows-CE steht auch diese Möglichkeit nicht zur Verfügung, da der CLR-Debugger keine Remote-Debugging-Unterstützung liefert.
- Komplexere Aufgaben, wie das Entwickeln eigener Steuerelemente- oder Klassenbibliotheken bleiben aufgrund der fehlenden Komfortfunktionen (Steuerelementedesigner, Organisation mehrerer Projekte in einer Solution, Debugging zur Entwicklungszeit, ...) dem Professional-Programmierer vorbehalten. Im Rahmen von Visual Studio stehen die genannten Funktionen dann zur Verfügung.

6.5 Übersetzungsmanager

Die Entwicklung einer mehrsprachigen Visualisierungsapplikation beinhaltet die Übersetzung aller in der Anwendung verwendeten Texte. Häufig wird dazu ein externer Übersetzer benötigt.

Als ”Quasi-Standard” zum Austausch der Texte zwischen Projekteur und Übersetzer beinhaltet VisiWinNET® Smart den Übersetzungsmanager. Hier lassen sich Excel-Listen erstellen, die die zu übersetzenden Texte aus dem VisiWinNET® Smart-Projekt enthalten. Nach dem Übersetzungsvorgang lassen sich die Inhalte der exportierten Dateien wieder in das Projekt einfügen.

Voraussetzung für den Übersetzer ist Excel aus dem Microsoft Office ab der Version ”Office 2000”.

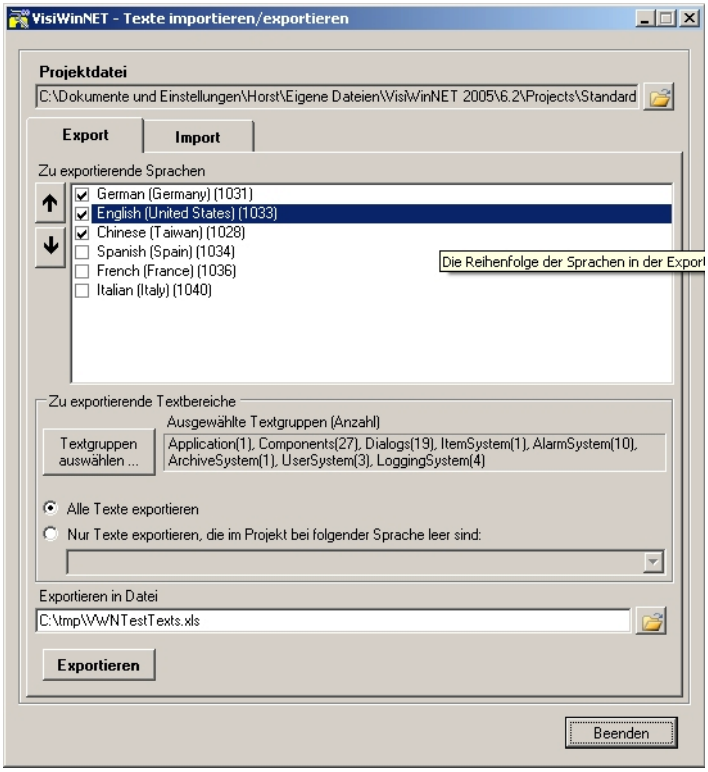
Der Übersetzungsmanager wird über das Windows Start-Menü gestartet. Bei der Installation von VisiWinNET® Smart wird ein entsprechender Eintrag in die Programmgruppe **VisiWin** → **VisiWinNET 2005** eingefügt.

Die Aufgabe des Projekteurs besteht darin, die zu übersetzenden Texte mithilfe des Übersetzungsmanagers als Excel-Datei zu exportieren, zu schützen und zum Übersetzer zu geben. Nach der Übersetzung muss der Projekteur die erhaltenen Informationen wieder importieren.

6.5.1 Texte exportieren

Der Export erstellt eine Excel-Datei, die die zu übersetzenden sprachumschaltbaren Texte aus dem Projekt enthält. Diese Datei kann an den Übersetzer gegeben werden.

Die Einstellungen und Funktionen für den Export befinden sich im Übersetzungsmanager auf der gleichnamigen Karteikarte.



Projektdatei: Dient zur Auswahl einer VisiWinNET® Smart-Projektdatenbank (*.vwn), aus der die Texte exportiert werden sollen. Über die nebenstehende Schaltfläche wird ein Dateiauswahldialog geöffnet.

Zu exportierende Sprachen: Listet die Sprachen des Projektes auf. Hier können maximal drei Sprachen für den Export selektiert werden. Die erste Sprache dient als Ausgangssprache der Übersetzung. Die zweite Sprache kann als Hilfssprache dienen. Die letzte Sprache kennzeichnet die Sprache, die vom Übersetzer auszufüllen ist. Über die Schaltflächen mit den Pfeilen wird die Reihenfolge der zu exportierenden Sprachen eingestellt.

Zu exportierende Textbereiche: Der Export kann nach einzelnen Textgruppen gefiltert werden. Der Auswahldialog für die Textgruppen wird über die Schaltfläche ”Textgruppen auswählen” geöffnet.

Der Auswahldialog beinhaltet folgende Bedienelemente:

- Über einen Klick auf das Selektionskästchen in der Gruppenansicht wird die Gruppe inklusive aller darunter liegenden Gruppen an-/abgewählt. Wird dabei die Strg-Taste gehalten, ändert sich der Selektionsstatus der Untergruppen nicht.
- Die Schaltfläche ”Alle” markiert alle gezeigten Textgruppen.
- Die Schaltfläche ”Keine” demarkiert alle Gruppen.
- OK bestätigt die Auswahl.
- Abbrechen schließt den Dialog, ohne die Auswahl zu übernehmen.

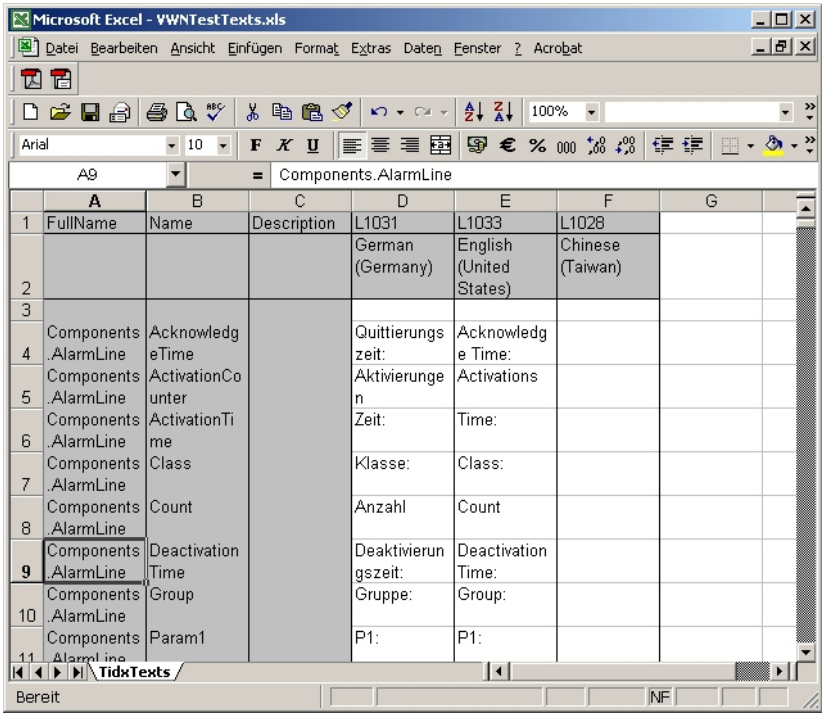
Alle Texte importieren/ Nur Texte importieren, die im Projekt bei folgender Sprache leer sind: Option für die wiederholte Übersetzung. Ist eine Sprache bereits einmal übersetzt worden, kann über die Option ”Nur Texte importieren, die im Projekt bei folgender Sprache leer sind” ausgewählt werden, das nur neue Texte (die also beim letzten Übersetzen nicht vorhanden waren) exportiert werden. Die darunter platzierte Liste bietet die Auswahl einer Sprache an, deren Inhalt auf Leertexte geprüft werden soll.

Exportieren in Datei: Gibt über einen Dateiauswahldialog (Schaltfläche) den Namen der neu zu erzeugenden Datei an.

Exportieren: Mit der Schaltfläche wird der Export gestartet.

Beenden: Schließt das Programm.

6.5.2 Exportdatei schützen



	A	B	C	D	E	F	G
1	FullName	Name	Description	L1031	L1033	L1028	
2				German (Germany)	English (United States)	Chinese (Taiwan)	
3							
4	Components.AlarmLine	AcknowledgeTime		Quittierungszeit:	Acknowledge Time:		
5	Components.AlarmLine	ActivationCounter		Aktivierungen	Activations		
6	Components.AlarmLine	ActivationTime		Zeit:	Time:		
7	Components.AlarmLine	Class		Klasse:	Class:		
8	Components.AlarmLine	Count		Anzahl	Count		
9	Components.AlarmLine	DeactivationTime		Deaktivierungszeit:	Deactivation Time:		
10	Components.AlarmLine	Group		Gruppe:	Group:		
11	Components.AlarmLine	Param1		P1:	P1:		

Die vom Übersetzungsmanager generierte Excel-Datei beinhaltet das Tabellenblatt ”Tidx-Texts” mit den exportierten Texten. Die ersten beiden Zeilen stellen den Tabellenkopf dar. Darunter folgen zeilenweise die Texte, wobei in den ersten beiden Spalten ”FullName” und ”Name” jeweils der für den Re-Import wichtige vollständige Pfad des entsprechenden Textes angegeben ist.

Die Excel-Datei beinhaltet einige Formalien, die nicht vom Übersetzer geändert werden dürfen.

- Der Tabellenkopf (die ersten beiden Zeilen) darf vom Übersetzer nicht verändert oder gelöscht werden.
- Auch für die ersten beiden Spalten gilt für den Übersetzer: Nicht ändern.
- Die Spalte ”Description” beinhaltet die ”Beschreibung” zum Text. Vom Projekteur sinnvoll ausgefüllt, kann sie eine wertvolle Hilfe für den Übersetzer sein. Der Inhalt wird aber beim Reimport ignoriert.
- Soll beim Re-Import die Option ”Nur Texte importieren, die in Projekt und Importdatei identisch sind” verwendet werden, darf der Übersetzer die Feldinhalte in der zum Vergleich verwendeten Sprache nicht ändern.

Hinweis!

Schnell ist eines der oben genannten Felder vom Übersetzer versehentlich geändert oder gelöscht. Sorgen Sie vor:

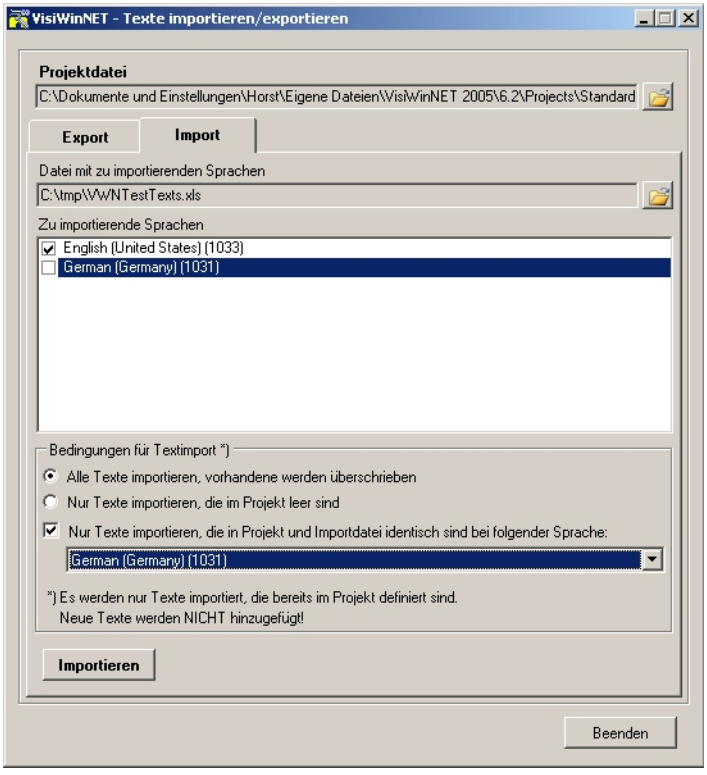
Excel bietet die Möglichkeit, Zellen eines Tabellenblattes zu sperren. Die entsprechenden Zellen sind dazu zuerst (zeilen- oder spaltenweise) zu markieren. Über das Menü ”Format”—>”Zellen” kann auf der Karteikarte ”Schutz” eine entsprechende Option aktiviert werden. Anschließend ist dem Tabellenblatt noch ein Kennwort zuzuordnen. Dadurch können auf dem Tabellenblatt nur noch die Felder geändert werden, die nicht gesperrt wurden. Weitere Informationen hierzu befinden sich in der Office-Onlinehilfe.

6.5.3 Texte importieren

Der Import dient dazu, die vom Übersetzer zurückgegebenen Texte aus der Excel-Liste in die Projektdatenbank zu übernehmen. Die Einstellungen und Funktionen für den Import befinden sich im Übersetzungsmanager auf der gleichnamigen Karteikarte.

Hinweis!

Der Import beschränkt sich dabei nur auf die in der Projektdatenbank bestehenden Texte. Wird die Excel-Tabelle um neue Zeilen erweitert, werden die hier so neu entstandenen Texte nicht in die Projektdatenbank eingefügt. Der Übersetzungsmanager ist also keine Alternative zum Editor der Sprachumschaltung. Möchten Sie Texte außerhalb des Editors in einer Tabelle vordefinieren und anschließend importieren, verwenden Sie stattdessen die Möglichkeit des Einfügens über die Windows-Zwischenablage. In den VisiWinNET® Smart-Editoren haben Sie die Möglichkeit, über die Funktionen ”Kopieren” und ”Einfügen” mit einem Excel-kompatiblen Zwischenablageformat zu arbeiten.


VisiWinNET - Texte importieren/exportieren
Projektdatei
C:\Dokumente und Einstellungen\Horst\Eigene Dateien\VisiWinNET 2005\6.2\Projects\Standard
Export
Import
Datei mit zu importierenden Sprachen
C:\tmp\VWNTestTexts.xls
Zu importierende Sprachen
English (United States) (1033)
German (Germany) (1031)
Bedingungen für Textimport *)
Alle Texte importieren, vorhandene werden überschrieben
Nur Texte importieren, die im Projekt leer sind
Nur Texte importieren, die in Projekt und Importdatei identisch sind bei folgender Sprache:
German (Germany) (1031)
*) Es werden nur Texte importiert, die bereits im Projekt definiert sind.
Neue Texte werden NICHT hinzugefügt!
Importieren
Beenden

Der Dialog hat folgende Bedienelemente:

Projektdatei: Dient zur Auswahl der VisiWinNET® Smart-Projektdatenbank (*.vwn), in das die übersetzten Texte importiert werden sollen. Über die nebenstehende Schaltfläche wird ein Dateiauswahldialog geöffnet.

Datei mit zu importierender Sprache: Dient zur Auswahl der Excel-Datei (*.xls), welche die vom Übersetzer zurückgegebenen Texte enthält. Über die nebenstehende Schaltfläche wird ein Dateiauswahldialog geöffnet.

Hinweis!

Die VisiWinNET® Smart-Systemtexte stehen in den Standardsprachen Deutsch und Englisch zur Verfügung. Zusätzlich finden Sie im Ordner "...\VisiWinNET 2005\Translations" Systemtexte in den Sprachen Französisch, Italienisch und Spanisch, die Sie über den Übersetzungsmanager in ihre Applikation einbinden können.

Zu importierende Sprachen: Wählt die Sprachen aus, deren Texte wieder in das Projekt zurückgelesen werden sollen.

Alle Texte importieren, vorhandene werden überschrieben: Gibt an, dass alle bereits im Projekt vorhandenen Texte in den ausgewählten Sprachen überschrieben werden sollen. Diese Option ist sinnvoll, wenn bereits eine Übersetzung in das Projekt importiert wurde und der Übersetzer nun weitere Anpassungen an den bereits importierten Texten vorgenommen hat.

Nur Texte importieren, die im Projekt leer sind: Gibt an, dass nur die leeren im Projekt vorhandenen Texte in der ausgewählten Sprache überschrieben werden sollen. Diese Option ist sinnvoll, wenn die im Projekt vorhandenen Texte bereits getestet wurden.

Nur Texte importieren, die in Projekt und Importdatei identisch sind bei folgender Sprache: Die unter dem Bedienelement liegende Auswahlliste bietet eine Sprache an, deren Texte miteinander verglichen werden. Nur wenn der Text dieser Sprache in externer Datei und im Projekt gleich sind, wird der entsprechende Text reimportiert. Üblicherweise wird hier die Ausgangssprache der Übersetzung angegeben. Ist z. B. nach dem Export an den Texten der Ausgangssprache weitergearbeitet worden, kann über das Setzen der Option verhindert werden, dass nun veraltete Übersetzungen mit in das Projekt importiert werden.

Importieren: Mit der Schaltfläche wird der Re-Import gestartet.

Beenden: Schließt den Dialog.

7 Stichwortverzeichnis

A

Alarm-Anzeigen filtern, 79

Alarm-Quittierungsart festlegen, 76

Alarme anlegen, 74

Alarme visualsieren, 77

Alarmgruppe anlegen, 73

Alarmsystem, 71

Alarmsystem testen, 78

Archivsystem, 80

Ausgabefenster, 10 , 20

Ausgeben von Projektdaten, 88

awl-Datei, 113

B

Bedienung, 9

Beispielprojekt, 33

Benutzertexte, 62

Benutzerverwaltung, 49

Browsen von Variablen, 40

C

CAN
- Kanal konfigurieren, 96 , 100
- PDO-spezifische Einstellungen, 101
- PDO-Variable anlegen, 101
- Variablen browsen, 97
- Variablen einlesen, 98
- Verbindung konfigurieren
  über Codestellen, 96
  über PDOs, 99

Clientbereich, 10

CoDeSys
- Kanal konfigurieren, 108
- Variablen einlesen, 111
- Verbindung konfigurieren, 108

D

Dongle, 6

Drucken, 88

Dynamische Texte, 68

E

Eigenschaftenfenster, 10 , 19

Eigenschaftenseite, 22

Eigenschaftenseiten, 10

Eingabegebietsschema, 66

EL 100, 7

Emulator
- mit Kommunikationstreiber, 27
- mit Variablensimulation, 27
- Windows CE, 26

F

Formulardesigner, 18

Formulare erstellen, 41

H

Hilfe, 11

HMI EL 100, 7

I

Informationen drucken, 88

Installation, 8

K

Kanal erstellen, 38

Kanal konfigurieren, 96 , 100 , 103 , 108

Kommunikationskanal erstellen, 38

Konfiguration, 91

Kontextmenü, 16

L

L-force Logic
- Kanal konfigurieren, 108
- Variablen einlesen, 111
- Verbindung konfigurieren, 108

Lieferumfang, 6

Lizenzbedingungen, 6

M

Menüleiste, 10 , 23

MPI
- Kanal konfigurieren, 103
- Variablen browsen, 105 , 106
- Verbindung konfigurieren, 103

P

PDO-spezifische Einstellungen, 101

PDO-Variable anlegen, 101

PLC Designer
- sym-Datei erstellen, 116
- Variablen browsen, 110

Programm
- beenden, 15
- installieren, 8
- Oberfläche, 9
- starten, 9

Projekt
- anlegen, 13 , 33
- Beispiel, 33
- Einstellungen, 91
- konfigurieren, 91
- konvertieren, 15
- mehrsprachig erstellen, 60
- öffnen, 12
- speichern, 14
- übertragen, 94

Projektexplorer, 16

Projektexplorer , 10

Prozessdaten aufzeichnen, 80

Prozessvariable definieren, 72

R

Rezepte, 84

S

sdf-Datei, 112

Sprache anlegen, 66

Spracheinstellungen, 93

Sprachumschaltung, 60

SPS-Programm, sym-Datei erstellen, 116

Steuerelemente einsetzen, 41

sym-Datei erstellen, 116

Symbolleiste, 10

Systemanforderungen, 7

Systemtexte, 133

T

Tabelleneditor, 21

Texte
- Benutzer-, 62
- dynamische, 68

Toolbox, 10 , 19

Trend, 80

U

Übersetzungsmanager, 127

V

Variablen browsen, 40 , 97 , 105 , 110

Variablen einlesen, 98 , 106 , 111

Visualisierung erstellen, 41

Visualisierungsrezepte, 84

W

Werkzeugleiste
- Format, 25
- Standard, 25

Werte darstellen, 41

Document history

Material number	Version			Description
13201079	1.0	03/2007	TD29	First edition
13227673	2.0	12/2007	TD29	Description of a project example added
13236341	3.0	04/2008	TD29	Description for the PLC Designer communication driver added
13292736	4.0	06/2009	TD29	Adapted to software version 6.3 and supplemented with the chapter ”Project example”
13342001	5.0	06/2010	TD29	Sections ”Scripts”, ”Restrictions for devices of the EL 100 series”, and ”Translation manager” supplemented
13351708	6.0	11/2010	TD29	Adapted to software version 6.4
13404124	6.1	03/2012	TD29	Adapted to software version 6.5

Tip!

Information and auxiliary devices related to the Lenze products can be found in the download area at

http://www.Lenze.com

1 Preface and general information 140
1.1 Scope of supply 140
1.2 Licencing terms 140
1.3 System requirements 141
1.4 Restrictions for devices of the EL 100 series 141

2 Installation 142

3 Operation 143
3.1 Basics 143
3.1.1 Start program 143
3.1.2 Description of the program's user interface 143
3.1.3 Help for the user 145
3.1.4 Open project 146
3.1.5 Create project 147
3.1.6 Save project 148
3.1.7 Convert project 149
3.1.8 Exit program 149
3.2 Integrated development environment 150
3.2.1 Project Explorer 150
3.2.2 Form designer, Toolbox and Properties window 152
3.2.3 Output window 154
3.2.4 Table editor 155
3.2.5 Property page 156
3.2.6 Menu bar 157
3.2.7 "Standard" toolbar 159
3.2.8 "Format" toolbar 159
3.3 Emulator for Windows® CE projects 160
3.3.1 Start emulator 161
3.3.2 Emulator window 162
3.3.3 Variable simulation window 164

4 Project example 168
4.1 Creating a project example 168
4.2 Creating a communication channel 173
4.3 Browsing variables 175
4.4 Creating a visualisation 176
4.4.1 Creating forms and displaying values 176
4.4.2 Creating other forms and navigating between them 180

4.5 User administration 184
4.5.1 Defining the type of user administration system 186
4.5.2 Creating rights 187
4.5.3 Creating user groups 188
4.5.4 Creating users 189
4.5.5 Using dialogues for user administration 191
4.6 Creating multiple-language projects (language switching) 194
4.6.1 Calling the language switching function 195
4.6.2 Creating user texts 196
4.6.3 Assigning user texts to a control element of the type ”Label” 197
4.6.4 Switching over the language during the runtime 199
4.6.5 Defining further languages 200
4.6.6 Using dynamic texts (showing variables) 202
4.7 Configuring the alarm system 205
4.7.1 Defining process variables 206
4.7.2 Creating alarm groups 207
4.7.3 Creating alarms 209
4.7.4 Specifying the type of alarm acknowledgement 211
4.7.5 Visualising alarms 212
4.7.6 Testing the alarm system 213
4.7.7 Filtering alarm displays 214
4.8 Recording process data (archive system) 215
4.9 Defining visualisation recipes 219
4.10 Printing information 223

5 Configuration 226

5.1 Projecte settings 226
5.1.1 Global project settings 226
5.1.2 Common project properties 226
5.1.3 Selection of the systems 226
5.1.4 Extended project properties 227
5.1.5 Runtime behaviour 228
5.1.6 Language settings 228
5.2 Communication 229
5.2.1 Transfer project via LAN to EL1xx 229
5.2.2 Configure CAN connection via codes 231
5.2.3 Configure CAN connection via PDOs 234
5.2.4 Configure MPI connection 238
5.2.5 Configuring the L-force Logic (CoDeSys) connection 242

6 Appendix 246
6.1 Generating an sdf file from a symbol table 246
6.2 Create instruction list file from a data block 247
6.3 Creating the sym file in the PLC Designer 250
6.4 Scripts 252
6.4.1 Basics 252
6.4.2 Troubleshooting within functions 257
6.4.3 Example functions 258
6.4.4 Restrictions in the programming interface of VisiWinNET Smart 260
6.5 Translation manager 261
6.5.1 Exporting texts 262
6.5.2 Protecting the export file 264
6.5.3 Importing texts 265
7 Index 267

1 Preface and general information

VisiWinNET® Smart is the powerful integrated development environment for operating units of the EL100 (HMIs) series or Industrial PCs (IPCs) from Lenze.

VisiWinNET® Smart is a tool with which you can create visualisation applications easily and comfortably.

In VisiWinNET® Smart, you can finally convert (compile) the applications into the data format specific to the operating unit in question and then transfer them to the corresponding operating unit.

1.1 Scope of supply

- DVD “VisiWinNET® Smart”
- Hardware dongle for licencing
- Licencing terms

Note!

After receipt of the delivery, check immediately whether the items match the accompanying papers. We do not accept any liability for deficiencies claimed subsequently.

Claim

- visible transport damage immediately to the forwarder
- visible deficiencies/incompleteness immediately to your Lenze representative.

1.2 Licencing terms

Both the integrated development environment and the runtime are subject to licencing terms and protected against misuse.

- Integrated development environment: Hardware dongle
- Runtime system: coupled to hardware

Note!

Without a hardware dongle, VisiWinNET® Smart can only be used in ”demo mode” with the following restrictions:

- Only projects with the preselected name ”VWNTest” can be created and edited.

In demo mode, the word ”Demo” is shown in the header.

1.3 System requirements

In order to be able to work with VisiWinNET®Smart, the hardware and software must, according to our experience, meet the following requirements:

Hardware	minimum	recommended
Computer	IBM-compatible PC	
CPU family	Intel Pentium/Celeron	
CPU clock frequency	500 MHz	1 GHz or higher
RAM	256 MB	512 MB or higher
Graphics	800 x 600 pixels, High Color (16 bit)	1280 x 1024 pixels, True Color (32 bit)
Free hard-disc space	200 MB for development system 280 MB for .Net-Framework (if not yet installed)	
DVD drive	Only for installation	
Ports	• Ethernet or CAN interface for communication with the target device (optionally free COM or USB terminal). • Free USB terminal for hardware dongle	

Software	
Operating system	Windows Server, XP SP2 or Vista
Applications	Microsoft .NET Framework (in the scope of supply if not already installed)
Optional applications *)	• Microsoft .NET Compact Framework 2.0 SP1, • Microsoft .NET Framework SDK 2.0 • PDF Reader (for user manuals) • Internet browser (for online help) • Microsoft ActiveSync.

*) Optional software products that are not available on the installation computer can be ordered free of charge via the Internet pages of the corresponding manufacturers. If these products are not installed, some advanced functions of VisiWinNET® Smart are not available. For installation, additional free memory space on the hard disk is needed.

1.4 Restrictions for devices of the EL 100 series

For performance reasons, in the case of HMIs of the EL 100 series there are the following restrictions in VisiWinNET® Smart :

Function	EL 106 ... EL 110s	EL 105 / EL 103 ECO
Maximum number of variables	1000	500
Minimum scanning time of a communication group	100 ms	200 ms
Maximum number of alarms	2000	1000
Alarm history, binary ring buffer	With a maximum of 1000 alarms	With a maximum of 500 alarms
Trends	• Max. 20 trends • Min. scanning time 500 ms • Max. 50 trend files during runtime • Min. cache size 10 • 2000 samples per file	• Max. 10 trends • Min. scanning time 1000 ms • Max. 50 trend files during runtime • Min. cache size 10 • 2000 samples per file
Logging	System is completely deactivated and cannot be re-activated, even by the user	
Recipes	• Max. 500 variables per recipe • Max. 50 recipe files during runtime	• Max. 250 variables per recipe • Max. 50 recipe files during runtime

2 Installation

Note!

Administrator rights are required for installation of the software!

Proceed as follows:

1. Insert the VisiWinNET® Smart DVD into your drive.

   If the auto-start function of your drive is active, the installation program starts automatically.

   If it has been deactivated, start the installation program by double-clicking the ”setup.exe” file in the Program directory of the installation DVD.

2. Follow the instructions of the installation dialogs. Go to the next dialog by clicking the **Next** button.

   As soon as VisiWinNET® Smart has been installed successfully, a message appears to confirm this. If errors occur during installation, the installation process is aborted. If this happens, contact Lenze Service.

3. Click the **Complete** button.

3 Operation

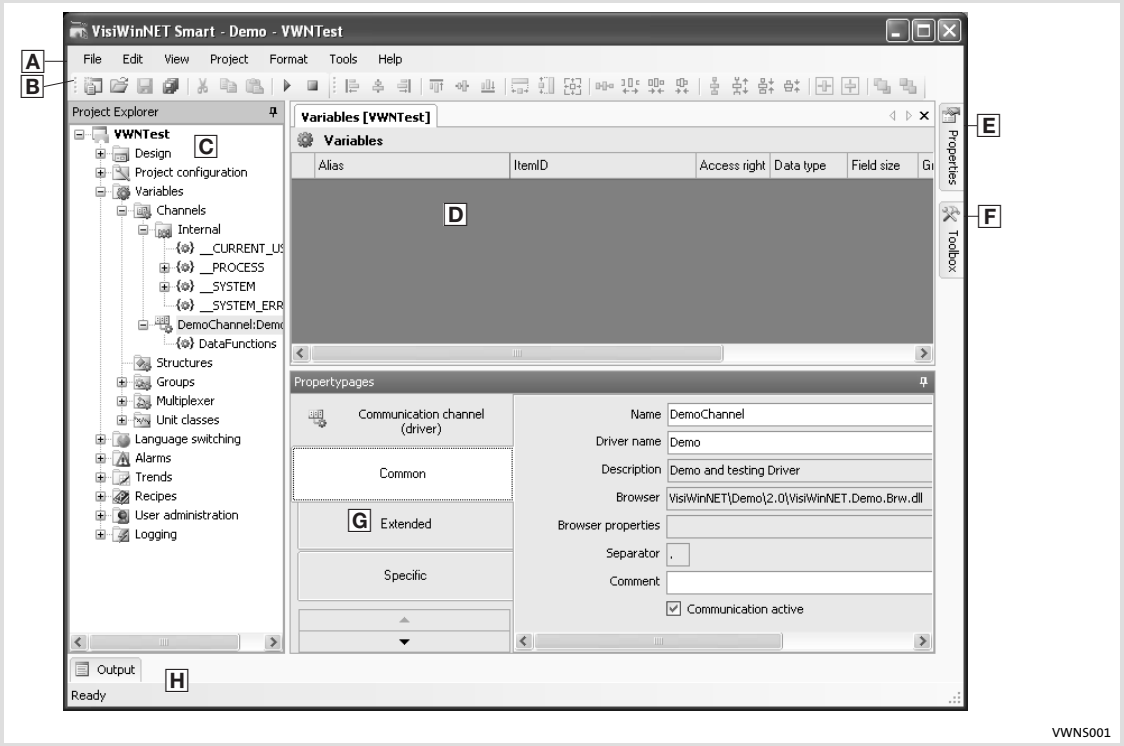
3.1 Basics

3.1.1 Start program

Proceed as follows:

1. Attach the hardware dongle.
2. **Open** the Start menu and click **All programs** → **Lenze** → **VisiWinNET 2005** → **VisiWinNET Smart**.
   The VisiWinNET® Smart screen is displayed.

3.1.2 Description of the program's user interface



A Menu bar
B Toolbar
C Project explorer
D Client area
E Toolbox window
F Property window (Tools)
G Property pages (object lists)
H Output window

Description of the user interface elements

- Menu bar A

  Each menu contains a series of menu commands which belong together thematically. When a menu command is selected, a function is started or a submenu is opened with further menu commands.

- Toolbar B

  The buttons of the toolbar can be used to call some functions quickly (without having to use the menus). The contents of the toolbar cannot be adapted by the user.

- Project Explorer C

  The structure and function of the project tree can largely be compared with WINDOWS Explorer. At the top (first level), there is the project folder. It contains (fixed) subfolders for structured object filing.

- Client area D

  In the client area, an object list or the selected form (runtime dialogue) are displayed, depending on the object selected in Project Explorer.

- Toolbox E

  The Toolbox contains the control elements for forms (designs) of the runtime system.

- Properties window ("Control elements" form) F

  Here, you specify the properties for the control element that has been selected in a form in the client area.

- Property pages G

  In the Property pages, you specify the properties of the object that has been selected in an object list in the client area.

- Output window H

  Display of system messages.

Note!

In various areas, you can click an object with the right mouse button to open a context menu that offers frequently used commands.

3.1.3 Help for the user

The user documentation for VisiWinNET® Smart includes the following components:

- Online Help

  VisiWinNET® Smart contains a detailed online help function. You can gain access to the online help system by means of the following:

  – Menu **Help** → **Helpe** (overall help system)

  – **Help** menu in a dialogue (context-related help)

  – By pressing the **[F1]** button, for example on a control element in a form, on a control element property (context-related help) or on a Property page for project definitions (channels, texts, alarms etc.)

  A precondition for calling the Online Help is that an Internet browser has been installed.

- User manuals (PDFs)

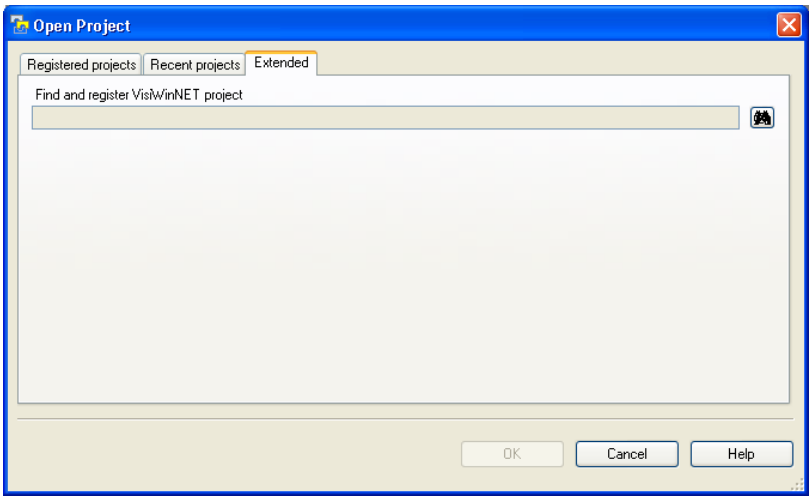
  All the information of the Online Help is also to be found in the PDF files under **Start** → **All programs** → **Lenze** → **VisiWinNET 2005** → **Documentation x.x [PDF]**.

3.1.4 Open project

Registering and opening a project

If you want to open a project that has not yet been registered at your workstation (i.e. has never been opened), proceed as follows:

1. Go to the **File** menu and click the menu command **Open project**.
   The ”Open project” dialogue is displayed.
2. Change to the **Extended** tab.

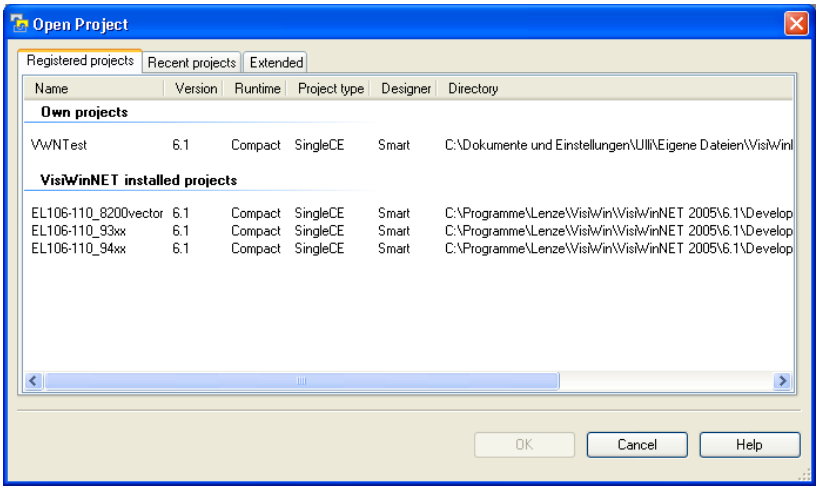


3. Click the **[...]** command button.
   The ”Open project” dialogue is shown.
4. Select the desired project folder.
   Standard: C:\Documents and Settings\Users\My Files\VisiWinNET 2005
5. Select the vwn file and then click the **Open** command button.
6. Close the ”Open project” dialogue by clicking the **OK** command button.
   The project will then be loaded.

Opening a registered project

If you want to open a project that has already been registered, proceed as follows:

1. Go to the **File** menu and click the menu command **Open project**.
   The ”Open project” dialogue appears.



2. In the **Registered projects** tab, click the project that you want to open.
   The project will then be loaded and displayed in Project Explorer.

3.1.5 Create project

In the chapter entitled ”Project example”, we explain how you can create a new project in VisiWinNET® Smart.(📖 168).

3.1.6 Save project

Save

Entries in the client area can be saved by clicking the menu item **File** → **Save** (entries in the current view) or **File** → **Save All** (all entries).

Note!

If you try to close a view or project that has not yet been saved, you are shown a reminder in the form of a dialogue. You can then accept the changes (**Yes**) button) or reject them (**No**) button).

Saving under another name

If you want to save the current project under a new name, select **File** → **Save Project As**. The changes to the original project are only saved in the new project; the original one remains unchanged.

Saving a project as a template

By selecting **File** → **Save Project as Template**, you can create a template that you can use as a basis for the creation of new projects. This is especially recommended if you frequently create similar projects and do not want to start from the beginning every time.

The management of project templates enables the user to call his own skeleton applications. It is permissible for project templates to contain completely configured forms as well as definitions.

Note!

If necessary, a dialogue entitled ”Select additional files” will ask you whether files that are not referenced from the project are to be copied into the newly created project directory. This especially means files such as HTML, XML, RTF or databases that are used by the application in control elements but have nothing to do with construction of a typical VisiWinNET SMART application.

3.1.7 Convert project

You must convert a project that was created in an older version of the program in order to be able to use all the advanced functions of the new program version.

Each version of VisiWinNET® Smart contains a fixed range of functions that are available in the projects created under the version.

By installing a new version of VisiWinNET, the project planner usually acquires a wider range of features that open up new possibilities. However, existing projects can continue to be opened and processed with the old range of functions.

Note!

- If you want to convert projects from an older version of the program into a newer version, both versions of the program must have been installed.
- You can process a project created in an older version of the program in a newer version of the program without having to convert the old project. Then, however, the advanced program functions of the new version of the program are not available to you.
- Conversion saves you the work of having to set up a project from the beginning when you change over to a new version of the program. However, it does not relieve you of the duty of thoroughly checking your application after conversion. Lenze does not provide a 100% guarantee of compatibility between the versions.

  Conversion automatically involves the risk of a change in the behaviour of your application!

1. Start the new version of VisiWinNET® Smart.
2. Open the project created in an older version of the program as described at the beginning.
3. Select the **File** menu, click **Convert project** and follow the instructions.
   A new project is created. The old project remains unchanged.

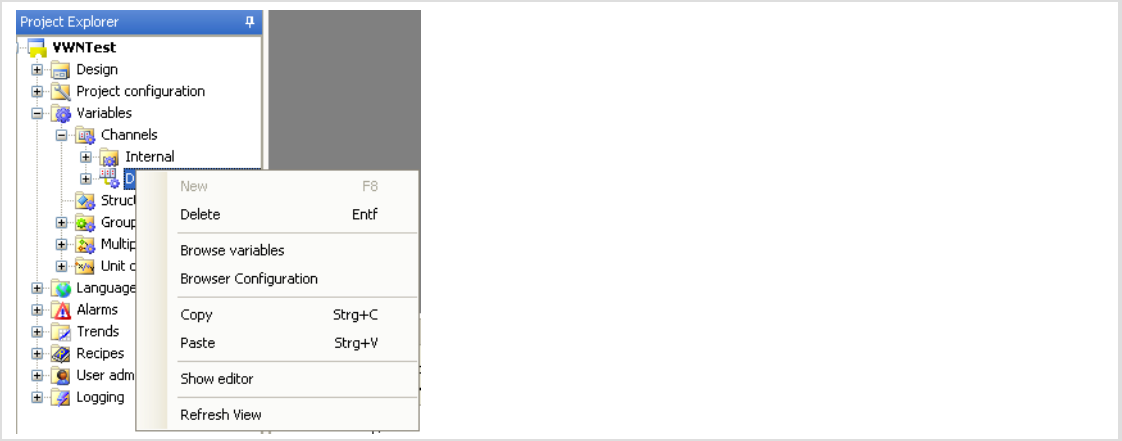
3.1.8 Exit program

Proceed as follows:

1. Go to **File** menu and click **Exit**.
   If you have made changes to the project and have not yet saved them, a dialogue appears listing all the changes. You can accept these changes (**Ja**) button) or reject them (**No**) button).
   VisiWinNET® Smart is then closed.

3.2 Integrated development environment

3.2.1 Project Explorer



Project Explorer enables access to the information in the project database and the forms of the opened VisiWinNET® Smart project. After the integrated development environment has been started, each node in the tree view of Project Explorer represents a system. Project Explorer also lists the forms of the project.

Project Explorer is used in a way comparable to Windows Explorer.

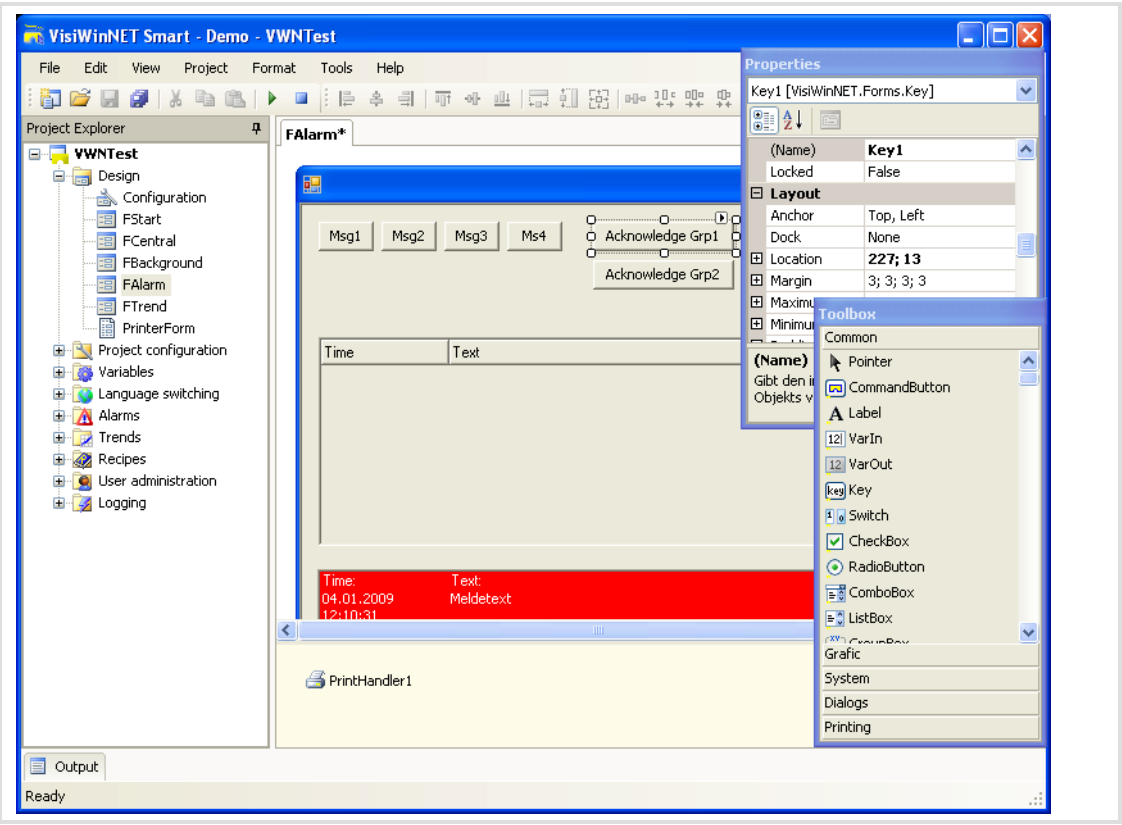
Context menu

Each system contains specific functions. These functions can be reached via the context menu of the corresponding node in Project Explorer. The context menu is opened by right-clicking the corresponding node.

Examples of context menus:

Nodes	Commands of the context menu	Function
Design	Add Form	Creates an empty form. The form dimensions from the project settings are used.
	Add Inherited Form	Creates a new form that inherits properties and control elements from a base form. The base form must be created in VisiWinNET Professional, must inherit from ”VisiWinNET.Forms.SmartForm” and must have been compiled into the project.
	Add Printer Form	Creates a printer form
	Add Template	Inserts an existing form from the template collection.
	Add Existing File	Inserts an existing form from another project.
	Add Folder	Creates a subfolder in the project directory
	Paste	Inserts the form from the clipboard under a new name.
Form	Open	Opens the detailed view of the selected form
	Exclude From Project	Removes the reference to the form from the project. The file is retained.
	Copy	Copies the form to the clipboard. It can be inserted into the project again under a new name by means of the ”Paste” context menu of the ”Design” node.
	Delete	Deletes the form from the project. Also deletes the form file.
	Rename	Changes the name of the form file
	Properties	Indicates the properties of the form in the Properties window.

3.2.2 Form designer, Toolbox and Properties window



The form designer is used to create and edit:

- ► screen views for the runtime system (🕮 176).
- ► printer forms for the output of information during the runtime (🕮 223).

It becomes visible if you double-click a form item in Project Explorer or if you select the ”Open” item from the context menu of the form item

When a project is created, the following three forms are created automatically:

FStart: This form initialises the runtime system. It must never be deleted under any circumstances!

FBackground: (only exists in Compact XP.) During the runtime, this form makes the desktop background go dark if you test applications on your PC with a lower resolution.

FCentral: This form is the central view of your runtime system. Starting from this form, you navigate to other forms and end the application.

Toolbox

The Toolbox contains the control elements that can be placed on the form. First, the desired control element has to be selected in the toolbox. Then, you left-click at the position on the form where the control element is to be placed. If the mouse button is released immediately, the control element is placed in this position and is of a previously specified size. If, in contrast, the mouse button is held down and the cursor is moved, the movement on the screen creates a frame. The effect of then releasing the mouse button is that the control element is created on the form and has the size of the displayed frame.

The position and size of control elements can be altered after the element has been placed in position. If a control element is selected with a mouse-click, a frame with pulling points appears around the control element.

How you move a control element:·

- ► Click the control element to select it.
- ► Hold the left mouse button down and drag the control element to the desired position.
- ► Release the mouse button.

How you alter the size of a control element:

- ► Click the control element to select it.
- ► Hold the mouse button down on a pulling point at the edges and pull it until the control element reaches the desired size.
- ► Release the mouse button.

Note!

You can move or scale several control elements at the same time by selecting them one after the other with the Ctrl key held down.

Typical help functions for aligning or positioning control elements can be called by means of the ”Format” toolbar.

Properties window

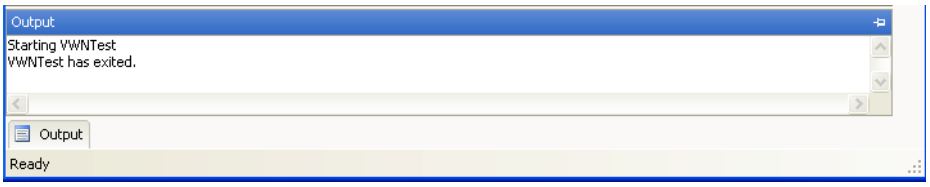
You specify the properties of a control element in the Properties window.

Here, you specify ”what” the control element is to show and ”how” the contents are to look. A typical example of ”what” is the specification of the process variable to be shown, whereby this variable is specified by means of the ”VWItem” property. An example of ”how” is the setting of colours by means of ”ForeColor” and ”BackColor”.

The Properties window always shows the properties of the selected control element. A control element is selected by means of a mouse-click as described above. The selected control element is highlighted by a frame.

If no control element has been selected, the properties of the form itself are shown.

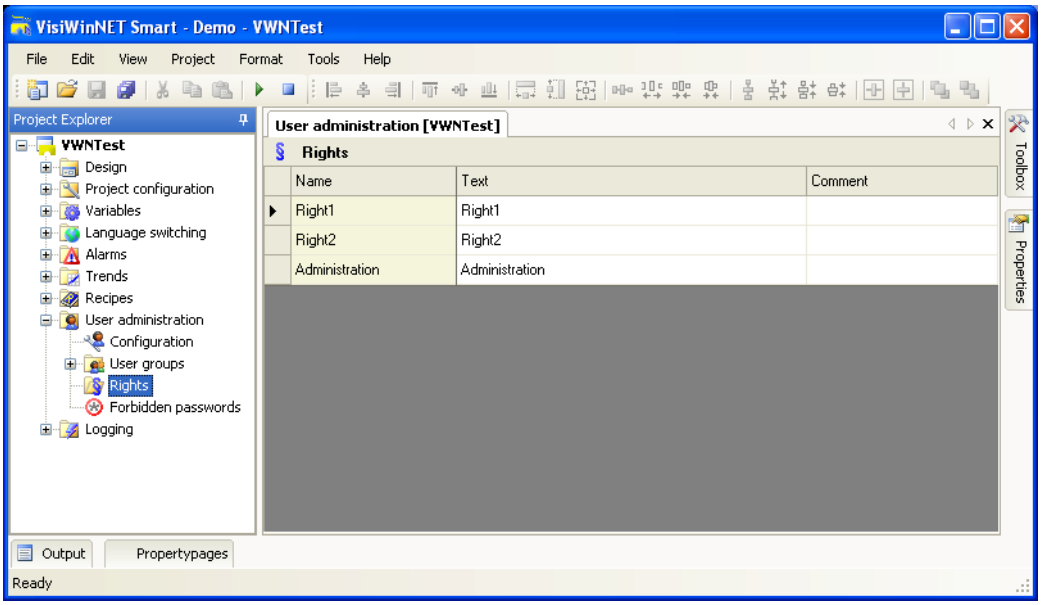
3.2.3 Output window



The Output window contains textual information on:

- exceptions that have occurred during the development time
- the progress of transfer to the target device

3.2.4 Table editor



In conjunction with Project Explorer, the table editors enable access to the definitions of the project database. Here, visualisation-specific definitions are made that, during the runtime, allow access to process data or describe special functions such as alarms, trends or language switching.

Each system of VisiWinNET®Smart is represented by a table editor and entries in Project Explorer. Clicking an entry in Project Explorer causes the table editor belonging to the entry to open. The table entries represent the definitions of the project database that are stored under the node highlighted in Project Explorer.

Each line is a definition. It contains the values of the parameters typical of the definition. If you select a definition, the associated Property page is displayed automatically. The context menu can be used to create new definitions or alter/delete existing definitions.

Note!

If you right-click a definition, the context menu opens, enabling you to create new definitions or duplicate, delete etc. existing ones.

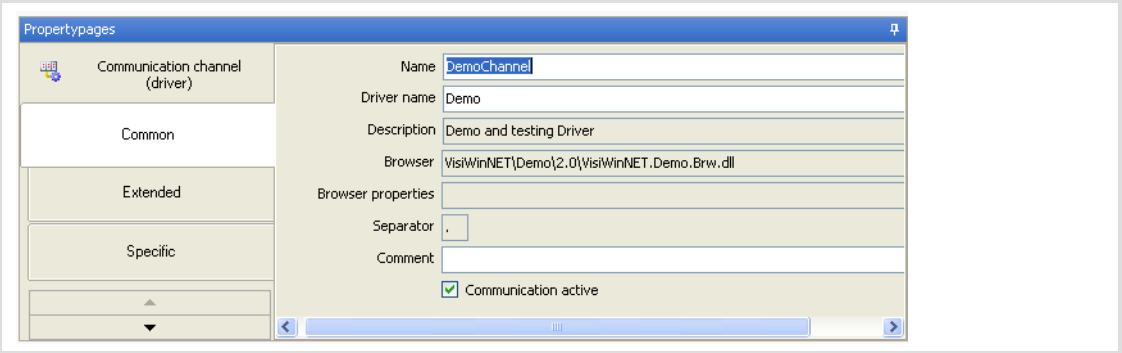
You can use the ”Options” command of the context menu to alter the way in which tables are shown. For example, you can specify the columns that are shown or the sequence of the columns in the table.

Some of the definitions are managed by Project Explorer itself. The definitions managed by Project Explorer are usually groups in which there can be further definitions. A typical example of this are the groups of the alarm system that are shown as a branch in Project Explorer. Subgroups and alarms can be configured in these groups. Subgroups that are created are shown in Project Explorer because they can also contain definitions. Alarms cannot contain any further definitions. They are therefore shown in the table editor.

Not all nodes enable the creation of definitions in the table editor. For example, trend definitions cannot be created under the Archive" node. Trend definitions can only perform a function in a higher-level group. In this case, this group would be an archive definition. First of all, therefore, an archive definition has to be created in Project Explorer. This is then to be selected in Project Explorer. Only then does it become possible to create trend definitions in the table editor.

Some specific groups are also blocked. They have already been created in a new project and cannot be deleted. The contents of this group cannot be extended in the table editor either. These groups contain definitions that are expected by the runtime system or by the control elements. Definitions in these groups can, however, be parameterised.

3.2.5 Property page

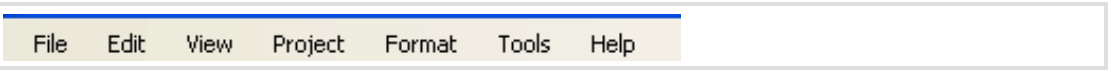


The Property page shows the parameters of the project database definitions selected in Project Explorer or the table editor.

Depending on the type of definition, the parameters are divided into different categories by means of tabs. The tabs in the area on the left-hand side of the Property page are used to change from one category to another.

An altered value is accepted if the cursor mark leaves the window (e.g. if another definition is selected). Only then do the altered values become visible in Project Explorer or the table editor.

3.2.6 Menu bar



The menu items in the main menu are used to perform general functions for the integrated development environment and for the design of forms. You can find other functions of VisiWinNET® Smart in Project Explorer and the table editors of the individual systems.

Menu	Menu command	Function	See also
File		Project and module management tasks	
	New Project	Creates a new VisiWinNET®Smart project.	📖 168
	Open Project	Opens a VisiWinNET®Smart project.	📖 146
	Close	Closes the view opened in the client area. You are asked to save the changes if you have made any.	
	Close Project	Closes the project currently open in the integrated development environment. You are asked to save the changes if you have made any.	
	Save	Saves the changes in an opened form.	📖 148
	Save Project As	Saves the project under a new name in any directory.	
	Save Project As Template	Saves the project as a user-defined template. When you create a new project, this project will be available as a user-defined template.	
	Save All	Saves all altered forms and project files.	
	Convert Project	Converts the loaded project to a newer version. **Note:** Conversion spares you the job of having to set up a new project when you change to a new version. However, it does not relieve you of the duty to thoroughly check your application again after conversion. Lenze does not guarantee 100%compatibility between the versions.	📖 149
	Exit	Ends VisiWinNET®Smart. Before exiting, you will be asked to save any changes to the project forms.	📖 149
Edit		General functions for altering definitions and designs	
	Undo	Undoes the last change.	
	Redo	Restores last undone change.	
	Cut	Cuts selected control elements out of the form and pastes them onto the clipboard.	
	Copy	Copies the selected control elements onto the clipboard.	
	Paste	Pastes control elements from the clipboard into the active form.	
	Delete	Deletes the selected control element.	
	Find	Looks for a word inside the active window. The search is started by means of a dialogue. Here, the following settings are possible: **Search in column:** This indicates the table column in which the search word is to be looked for. **Search word:** This indicates the character string to be searched for. The search word is limited to the possible input values of the column to be searched through. **Case-sensitive:** This ensures the search takes upper case and lower case letters into account. **Search entire column:** If this option is active, only those fields are shown whose contents completely match the search word. The search procedure is started by clicking **Find**.	
	Find Next	Continues the search from the current position.	
	Replace	Replaces the search word with a new text.	

Menu	Menu command	Function	See also
	Select all	Highlights all the definitions or objects in the active window.	
View		Activation of windows and toolbars	
	Output	Causes the output window to appear	📖 150
	Properties	Causes the Properties window to appear	
	Project Explorer	Causes the Project Explorer window to appear	
	Toolbox	Causes the Toolbox window to appear	
	Toolbars -> Format	Causes the ”Format” toolbar to appear or disappear in the toolbar area. Visibility is indicated by a checkbox in the menu.	
	Toolbars -> Standard	Causes the ”Standard” toolbar to appear or disappear in the toolbar area. Visibility is indicated by a checkbox in the menu.	
Project		Using the runtime system	
	Start	Starts the runtime project.	
	Exit	Stops the current instance of the runtime project.	
	New Project	Creates a runtime project.	
	Transfer To Target Device	Only under Windows XP or Windows XP Embedded: Transfers the runtime project to a target device without starting the project. A connection dialog is displayed if a connection has not yet been established. (See ”Connect to target device” in the Extras menu.)	📖 229
	Start On Target Device	Starts the runtime project on a target device. A connection dialog is displayed if a connection has not yet been established. (See ”Connect to target device” in the Extras menu.)	
Format		The ”Format” menu contains functions that can be used on control elements in forms in order to scale or position them, for example.	
Tools		Connecting to the target device	
	Connect To Target Device	Establishes a connection to a ”Windows CE” device. At the opposite end, the VisiWinNET®Smart RemoteAccessManager is expected.	
	Disconnect Target Device	Breaks off the connection to the ”Windows CE” target device. This menu is only available in the case of ”Compact” projects for Windows-CE.	
	Options	Contains the following setting possibilities, among other things: **Common:** Specifies the number of projects that are to be displayed in the current list when the ”Open Project” command is executed. **Directory for my projects:** Specifies the directory to be the set directory for a new project that is created. **Directory for my templates:** Directory in which your own templates are stored. Forms can be selected from this directory to serve as templates in other projects.	
Help		Call help and support information	
	Content, Index, Quick Start	Calls the online help system.	
	Start netviewer session		
	About ...	Calls a dialogue that contains version information on the installed VisiWinNET®Smart products.	

3.2.7 ”Standard” toolbar

Frequently used commands from the menus **File**, **Edit**and **Project**have been assigned to the command buttons of the format toolbar. If you place the cursor on an icon, a tooltip is shown with the menu command.

The ”Standard” toolbar is made visible by selecting the menu command **View** → **Toolbars** → **Standard**.

3.2.8 ”Format” toolbar

Frequently used commands from the **Format** menu have been assigned to the command buttons of the format toolbar. If you place the cursor on an icon, a tooltip is shown with the menu command.

The Format toolbar is made visible by selecting the menu command **View** → **Toolbars** → **Format**.

3.3 Emulator for Windows® CE projects

From version 6.5 onwards, **VisiWinNET® Smart** includes an emulator for Windows® CE projects.

If, during the development of Windows® CE projects, no target device is available, this emulator can be used to test a visualisation on the development PC.

Note!

Please note that this is only a pre-test during the development phase. It does not replace the final tests on the target system.

Functional differences between emulation and real runtime system:

- Data link

  All communication components (and thus the real connection to the PLC) are simulated. Via the interface of the emulator, variables can be set during simulation.

- Framework

  The emulator uses the Microsoft® .NET Standard Framework. Hence, functional differences may occur if functions of the Microsoft® .NET Compact Framework are used under Windows®CE.

- Performance

  Charging times or time flows of the final runtime system considerably depend on the power of the target device.

3.3.1 Start emulator

Note!
The emulator can only be started from a loaded Windows® CE project.

You can start the emulator with the following options:

- Emulator with variable simulation
  Simulates the access to PLC variables.
- Emulator with active communication
  The communication driver establishes a real connection to the projected PLC. It supports the following Lenze drivers:
  – LogicAndMotion V3
  – CoDeSys
  – S7 TCP/IP
  Unless supported communication components are projected, these will be deactivated when the ”Emulator (with active communication)” is started.

How to proceed:

1. Open the combination field in the toolbar area and select the desired option.

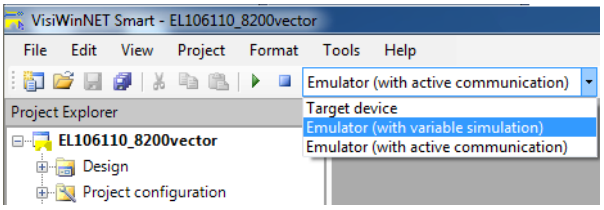


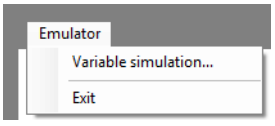
Fig. 3-1 Select emulation variant

2. Compile the project as usual and start, e.g. with [F5].
   The emulator window with the application interface is opened.

3.3.2 Emulator window

Fig. 3-2 Example for an emulator window

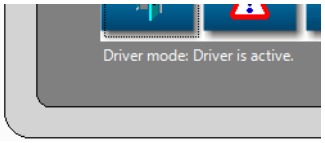
Emulator menu



Variable simulation: Open variable simulation window.

Exit: Exit emulator.

Driver mode

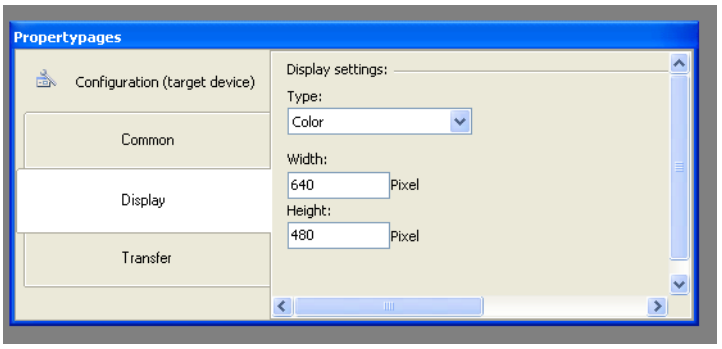


Driver mode: Simulation: Emulator with variable simulation is active. The access to PLC variables is simulated.

Driver mode: Driver is active: Emulator with communication driver is active. The project establishes an active connection to the projected control in the emulator. Hence, the variable values are not simulated but stem from the control.

Window size

The window size is defined in the object tree under **Project configuration** → **Target device** → Index card **Display**.



Note!

The emulation window can maximally adapt to the resolution of the target system. The representation of the application is scaled or cut depending on the ”ScaleToScreenSize” feature. Thus, it may differ from the representation on the target device.

3.3.3 Variable simulation window

Call: Click ”Variable simulation” in the menu of the emulator window.

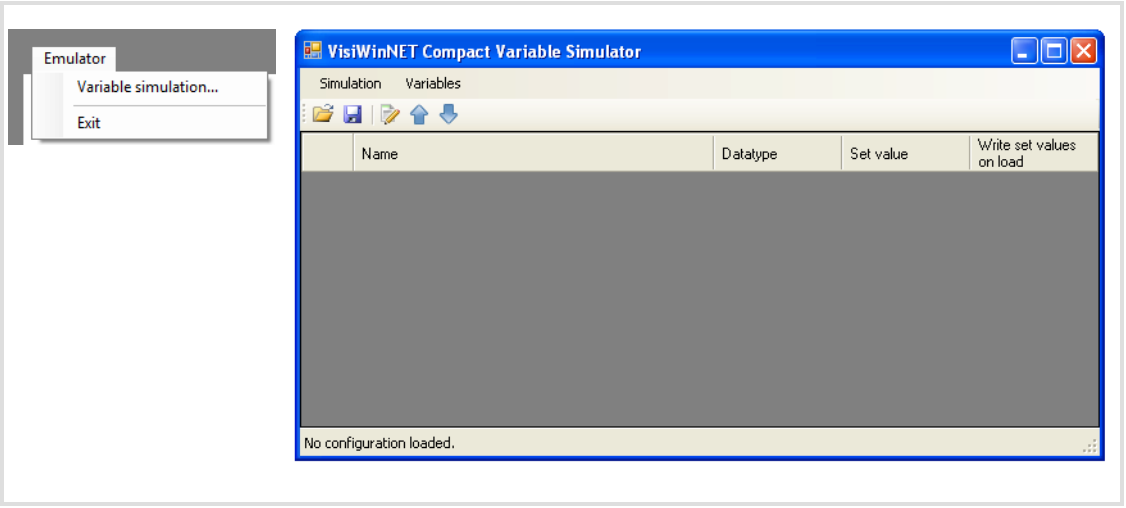
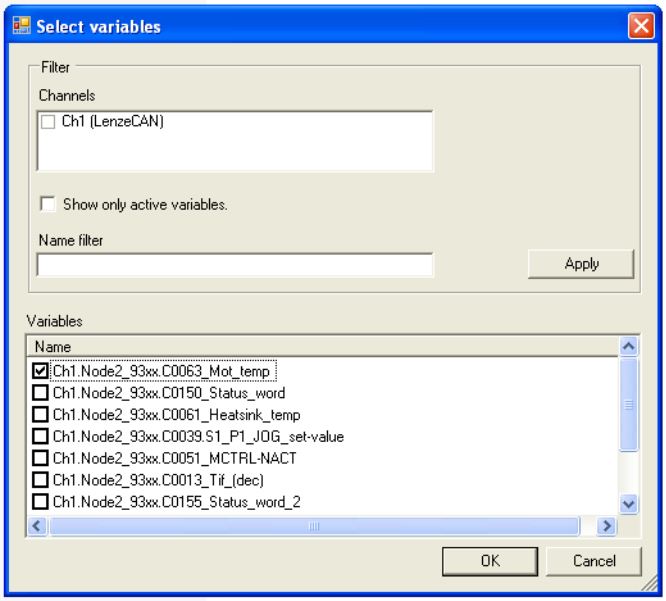


Fig. 3-3 Variable simulation window without loaded configuration

Select variables

Call: Click **Variables** → **Select variables** in the variable simulation window.



”Filter” area

Channels: Select a channel, the variables of which are to be displayed in the lower window.

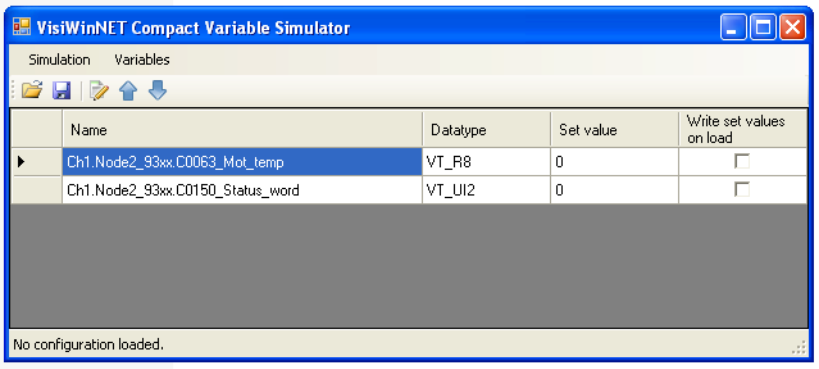
Show only active variables: If marked, only active variables are displayed in the lower window.

Name filter: Define filter for variable display. Case-sensitive. Wildcards (”*”, ”?” etc.) are not permissible.

Variables: Select the variables the values of which are to be simulated.

[OK]: Close window. The selected variables are displayed in the list field of the variable simulation window.

Define setpoints



List field

Name: Name and address of the variables

Data type: Type of variables (read only)

Set value: Enter the desired value of the variables here.

Write set values on load: If marked, the value is directly written while the configuration file is being loaded.

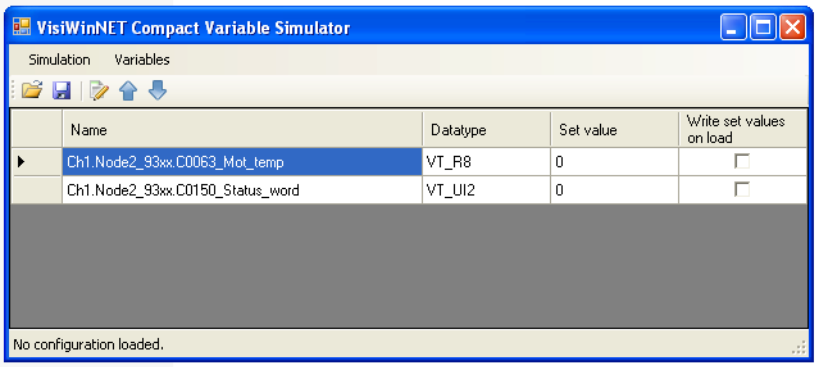
”Variables” menu”

Select variables: Open the ”Select variables” window

Write set values: Transmit variable values to the variable core. Then the current value switches and displays the setpoint.

Read current values: Read variable values from the variable core.

Create and load configuration file



The of the variables shown in the variable simulation window including their values can be saved in a file. For later emulation runs, the saved variable lists can be loaded.

”Simulation” menu

Load configuration: Load existing configuration. The currently loaded configuration is displayed in the lower window.

Save configuration/Save configuration as: Save current variable selection as configuration.

Delete configuration: Delete currently loaded configuration.

Set initial configuration: Declare currently loaded configuration as initial configuration.

When the project is restarted in the emulator, the initial configuration is loaded automatically. All set values with an active ”Write set values on load” option are directly written into the variable core. Set values where that option is not active will ony be written when the ”Write set variables” function is called from the ”Variables” menu.

Note!

We recommend to define a configuration file for each function block (form or part of a form).

4 Project example

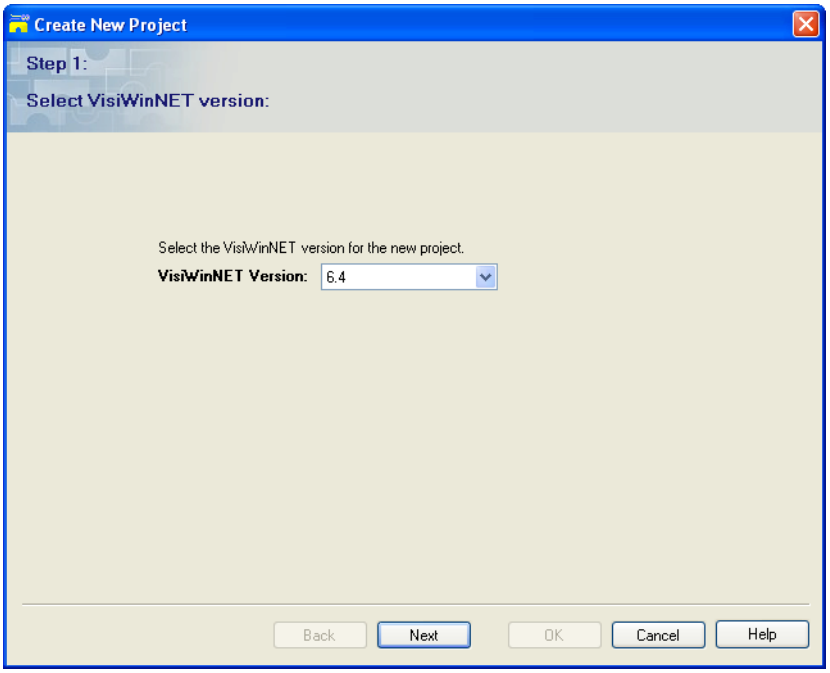
This chapter uses examples to describe the basic procedures for creating a visualisation application.

If you work through the prescribed steps, you will become familiar with the main functions of VisiWinNET® Smart quickly and efficiently.

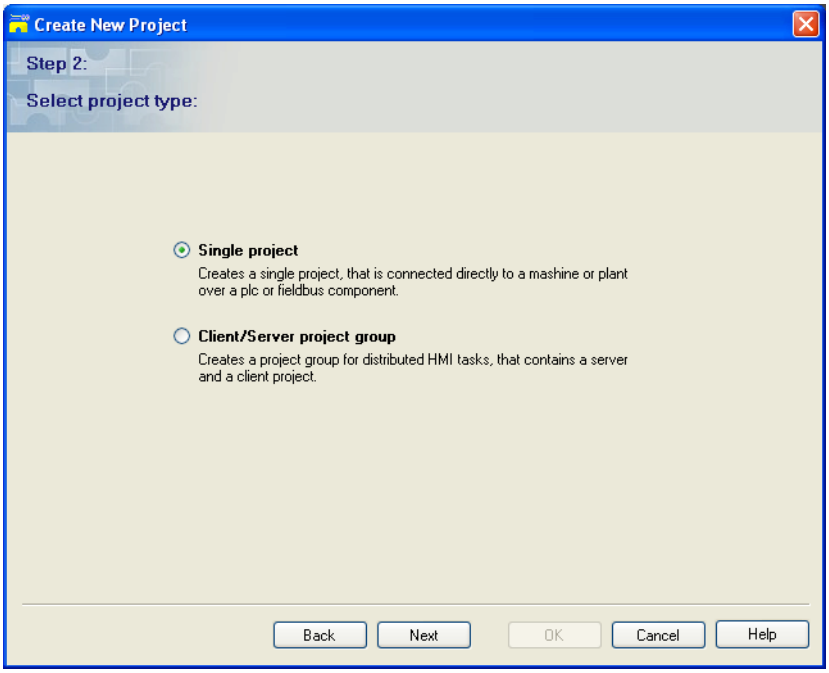
The example is based on the VisiWinNET® Smart demo driver. This means that you do not need an external target system. You can therefore work through the steps described below on your desktop computer or laptop at your own speed as long as you have installed VisiWinNET® Smart there.

4.1 Creating a project example

1. In VisiWinNET® Smart, click **File** → **New Project**.
   In the following dialog, you can specify the version of the program for which you want to create the project. (📖 149).

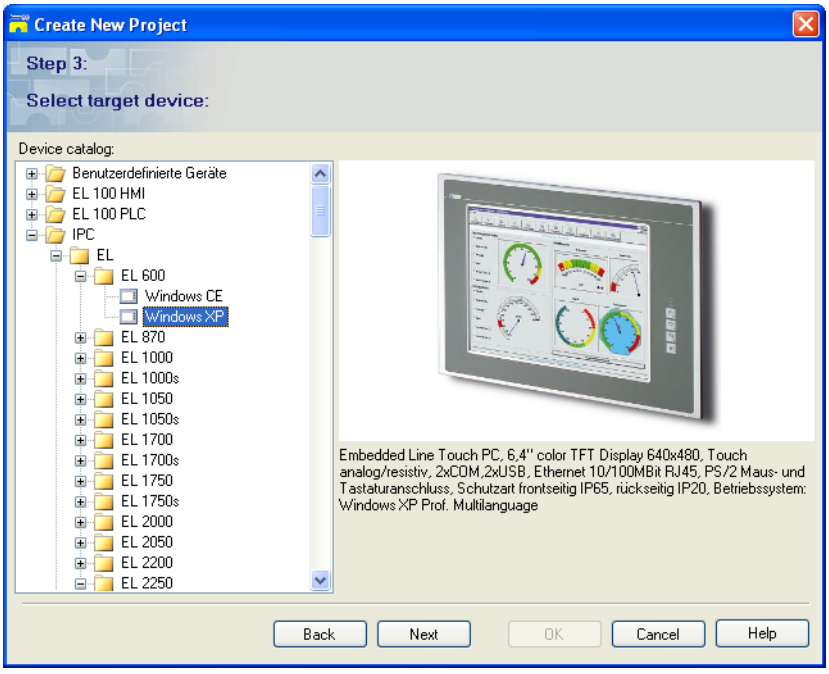


2. Click **Next**.
   In the following dialog, you can choose whether you want to create a single project or a client/server project.

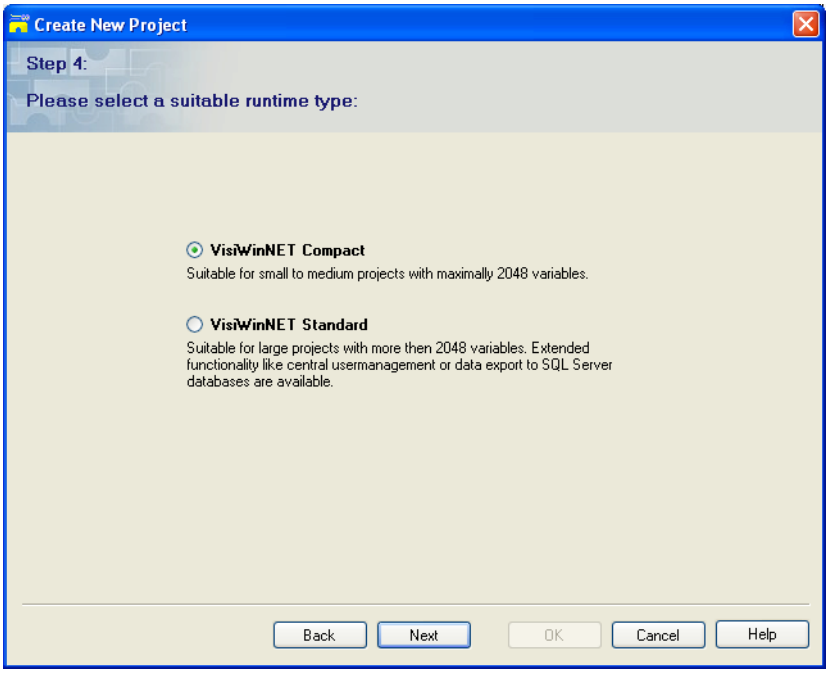

Create New Project
Step 2:
Select project type:
Single project
Creates a single project, that is connected directly to a mashine or plant over a plc or fieldbus component.
Client/Server project group
Creates a project group for distributed HMI tasks, that contains a server and a client project.
Back
Next
OK
Cancel
Help

Below, we describe how a single project is created.

3. Select **Single project** and then click **Next**.
   In the following dialog, select the target device for this project.

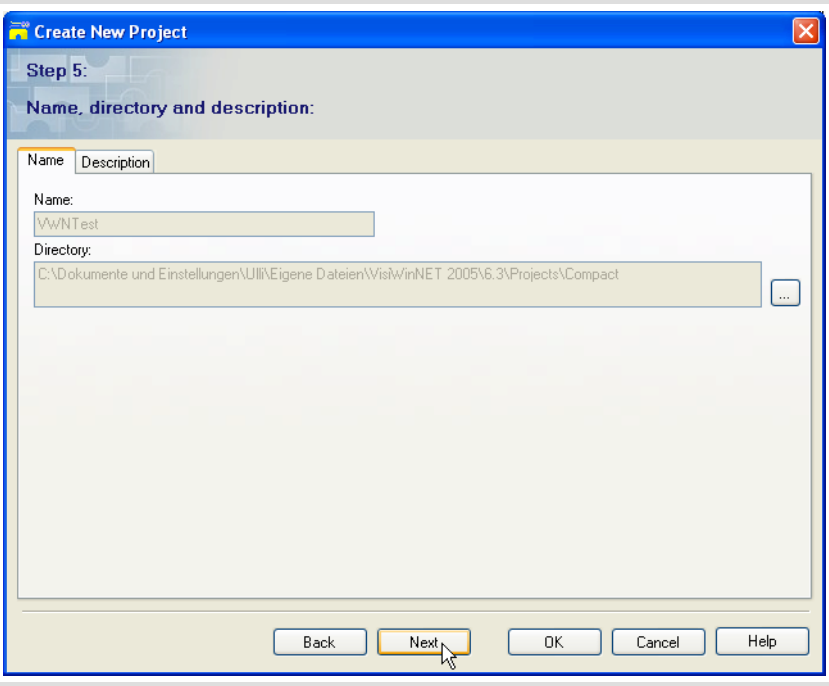


4. Because we are using the demo driver to set up the project we are using as an example, you can choose any **IPC**.
5. Select the **Windows XP** operating system and then click **Next**.
   In the following dialog, you can choose whether you want to create a ”Compact” or a ”Standard” project.

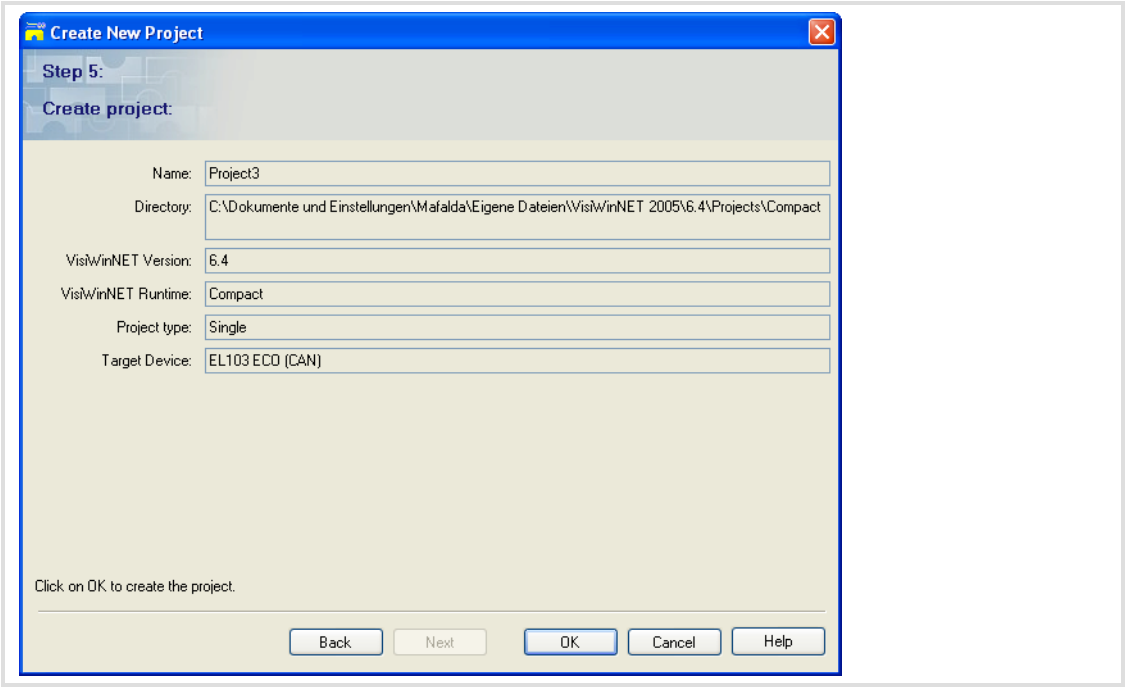

Create New Project
Step 4:
Please select a suitable runtime type:
VisiWinNET Compact
Suitable for small to medium projects with maximally 2048 variables.
VisiWinNET Standard
Suitable for large projects with more then 2048 variables. Extended functionality like central usermanagement or data export to SQL Server databases are available.
Back
Next
OK
Cancel
Help

Below, we describe how a ”Compact” project is created.

6. Select **VisiWinNET Compact** and then click **Next**.
   In the following dialog, you can specify a name and a storage location for the project.
   In the demo mode (without dongle), you cannot change the preselected project name, which is ”VWNTest”.

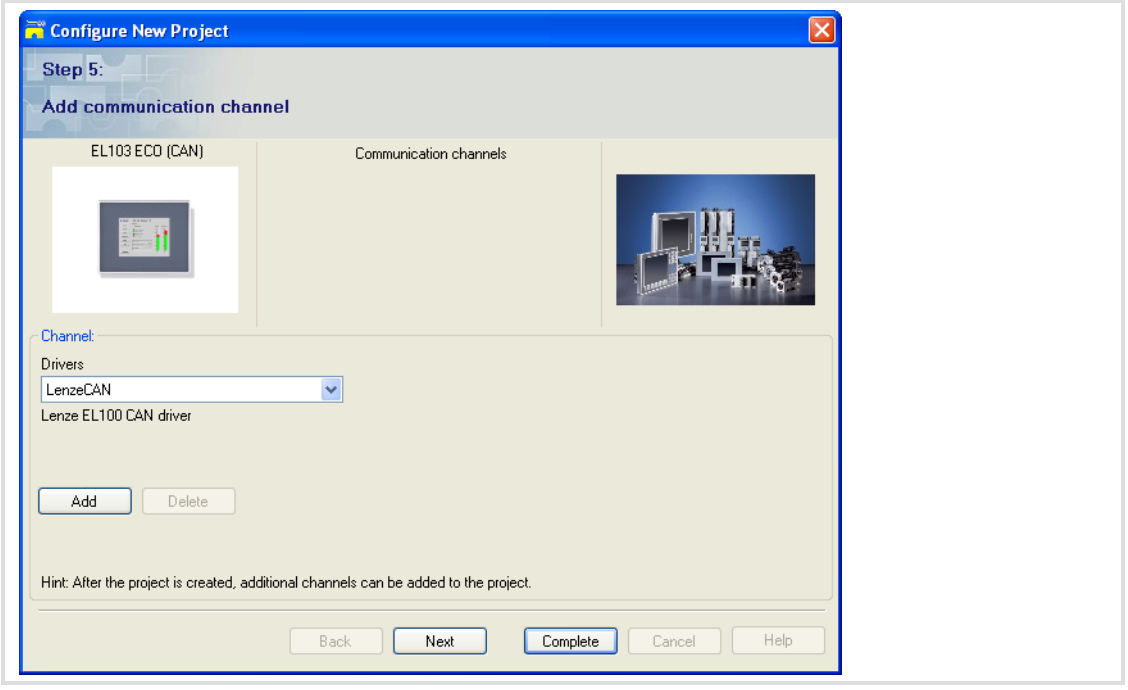


7. Click **Next**.
   The following dialog provides an overview of all project settings.



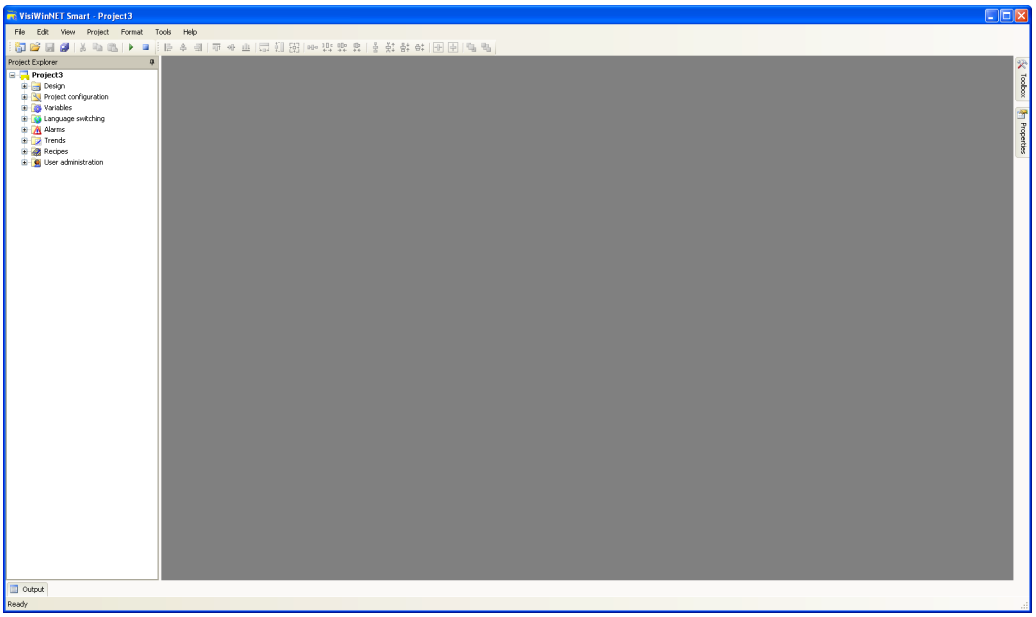
8. Click **OK** in order to create the project.

   In the following dialogues, you can create and configure a communication channel and also specify other project data. At this point, we want to skip this for the time being. We describe an alternative procedure in the next section.



9. Click **OK**.

   The project is created and displayed in Project Explorer.


VisiWinNET Smart - Project3
File
Edit
View
Project
Format
Tools
Help
Project Explorer
Project3
Design
Project configuration
Variables
Language switching
Alarms
Trends
Recipes
User administration
Toolbox
Properties
Output
Ready

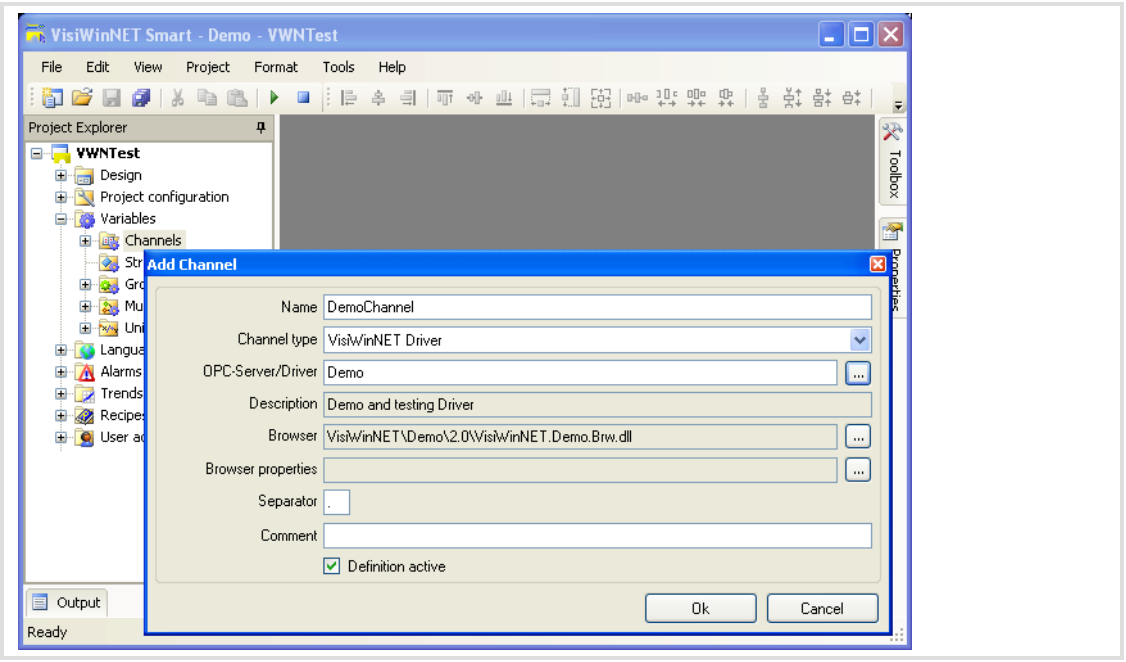
4.2 Creating a communication channel

The definition of a communication channel specifies the communication component (e.g. HMI 100 or IPC) with which data are to be exchanged. Here, we use the demo driver to create a communication channel.

1. In Project Explorer, open the **Variables** node and select the **Channels** node.
2. Click the **Channels** nodewith the right mouse mutton and, in the context menu, select **New**. As an alternative, you can press the **[F8]** function key.

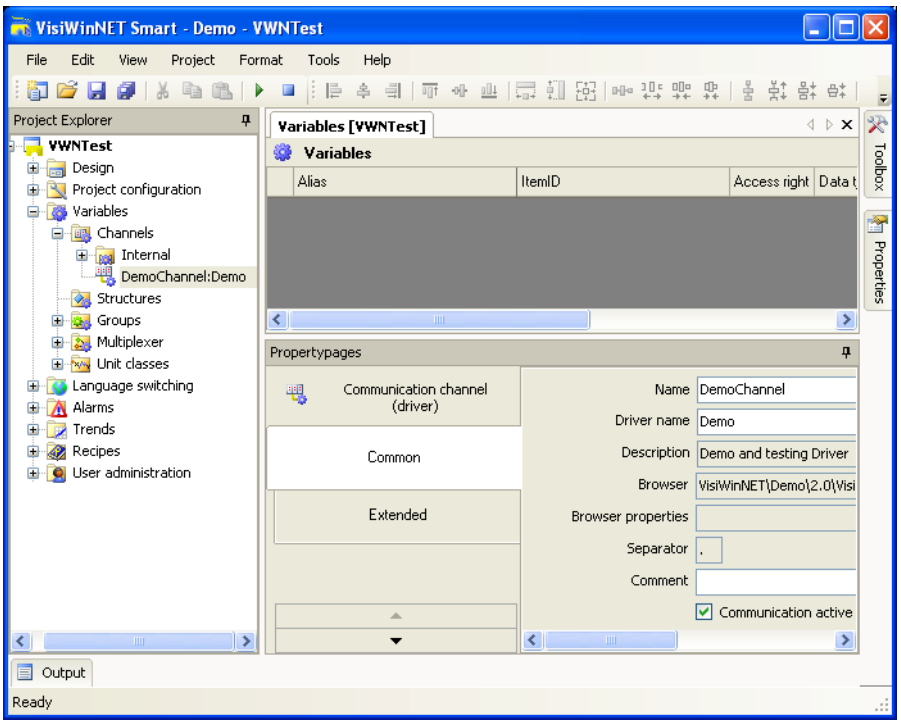
   The ”Add channel” dialogue then opens.
3. Enter **Demo channel** as the name. Then click the **[...]** command button next to the field ”OPC-Server/Driver” and, in the selection dialogue, select **Demo driver.**

   After you have closed the Selection dialogue by clicking **OK**, the appearance of the ”Add channel” dialogue should be as follows:



4. Click **OK**.

   The channel is then created and displayed in Project Explorer.



5. In order to check whether the communication channel is working correctly, open **Project** menu and click on the **Start** menu item.

   (Alternatives: Press **[F5]** or click the command button ▶ in the ”Standard” toolbar)

   If the communication channel is working perfectly, a VisiWinNET® graphic is briefly shown and then a visualisation form appears with an ”End” command button.

Note!

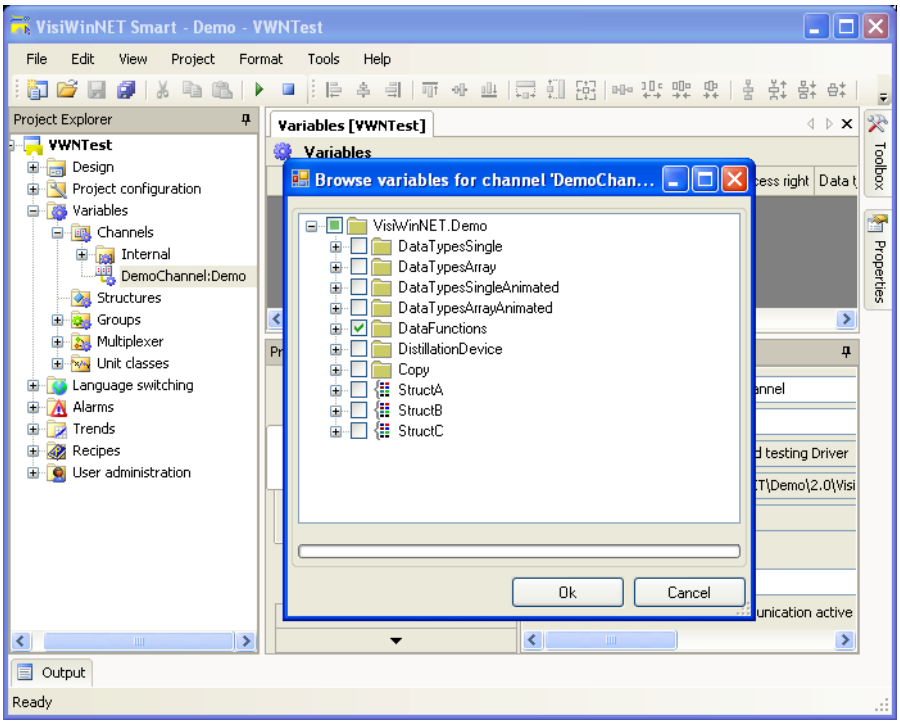
If you are working in demo mode (without dongle), an information dialogue appears every time the runtime system is started. Close it by clicking the **OK** button.

4.3 Browsing variables

This section describes the way in which variable definitions from project files, the PLC or other sources are read in. For our project example, we choose a source that supplies us with examples of values such as a saw-tooth curve. You can find more examples of variables browsing in chapter ”5.2 Communication”. ”5.2 Communication”.

1. Go into Project Explorer and open the node **Variables** → **Channels**.
2. Click **Demo channel** and, in the context menu, select the **Browse variables** command.

   The dialog ”Browse variables for channel ...” opens.



3. Select the **DataFunctions** control field and click **OK**.

   The entry ”DataFunctions” appears under the ”DemoChannel” object in Project Explorer.

4.4 Creating a visualisation

This section describes the general use of forms (designs) and their control elements.

A form represents a screen view or a dialog. It visualises the process variables from the project database and contains control elements, e.g. for navigation between different forms.

4.4.1 Creating forms and displaying values

After creation of a new project, some forms have already been created.

FStart: This form initialises the runtime system. It must never be deleted under any circumstances!

FBackground: This form darkens the desktop background during the runtime if you test applications with a lower resolution on your PC.

FCentral: This form is the central view of your runtime system. Starting from this form, you navigate to other forms and end the application.

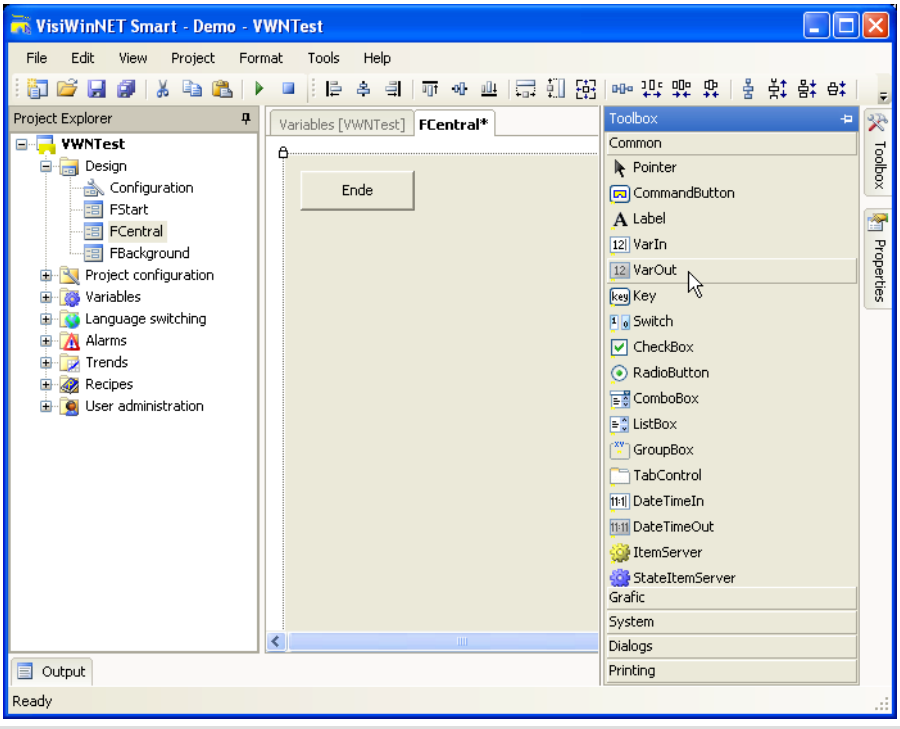
1. Go into Project Explorer, open the **Design** node and double-click the form object **FCentral**.

   The form represents a screen view with a control element of the type ”CommandButton” (End).

   The appearance of the screen view is determined by the way in which control elements are placed and parameterised. Control elements are selected in the Toolbox.

2. Open the Toolbox by clicking the **Toolbox** tab at the top right-hand edge of the window.

3. Under **Common**,select the control element type **VarOut**.



4. Hold down the left mouse button and drag the control element from the Toolbox onto the form.

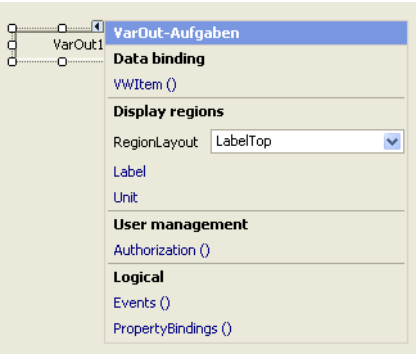
   The control element is placed and is of a predetermined size.

Tip!

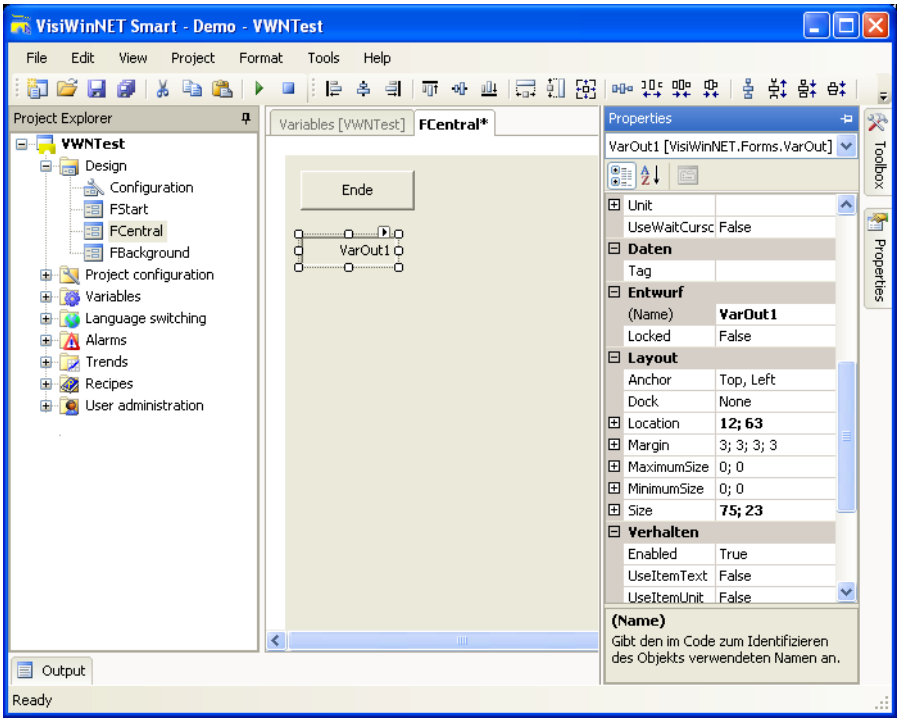
In the standard setting, the Toolbox window is closed automatically as soon as you click anywhere outside the Toolbox. You can alter this behaviour by clicking the pinboard pin icon in the top right-hand side on the Toolbar:

- ► If the tip of the pin is pointing towards the left, the Toolbox is closed automatically.
- ► If the tip of the pin is pointing downwards, the Toolbox remains open.

The associated tasks menu is opened at the same time as the control element is placed on the form.



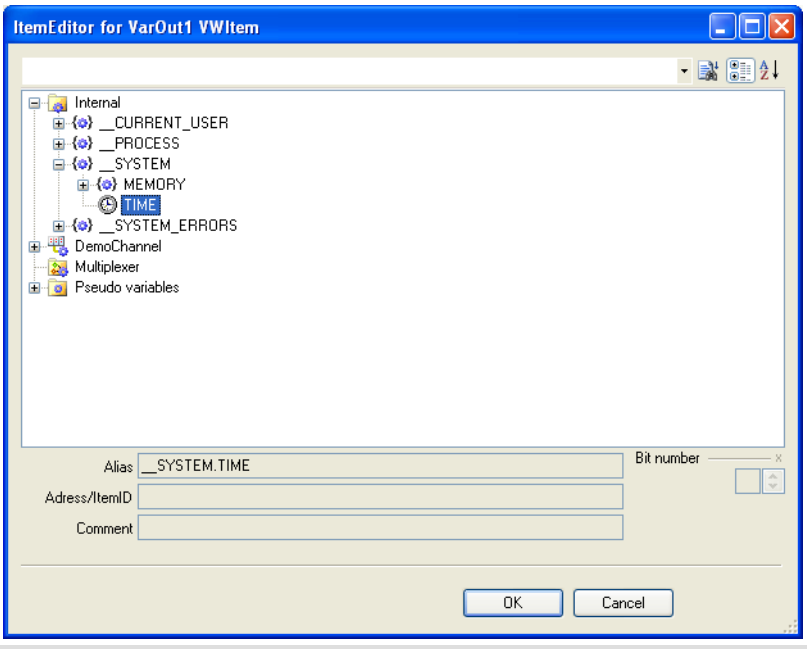
5. Close the tasks menu by left-clicking somewhere outside the menu.
6. Select the new control element and open the associated Properties list by clicking the **Properties** tab at the top right-hand side of the window.



The Properties window shows the current parameters of the control element.

The selected type of control element is used for the numerical or alphanumerical display of variable values. The most important aspect to be defined is the process value that is to be shown. This is done by means of the ”VWItem” property (at the end of the list).

7. In the Properties window section **VisiWinNET**, select the property **VWItem** and click the **[...]** command button.



The dialog shows the available process variables that have been stored in the project database as a result of browsing or self-definition. The structure of the hierarchy shown here is basically the same as the structure in Project Explorer.

”Internal” indicates variables that have not been assigned to any communication channel, and that are not mapped realistically in a control system. Nevertheless, these variables can be needed within a visualisation application in order to trigger e.g. variable-controlled processes or can simply be needed as a buffer for values from the visualisation application.

”DemoChannel” or ”External”contains the variables that are mapped in the PLC. The communication channels are listed under ”External”. Under this area are the name spaces that have already been using during browsing.

8. Select the variables whose value you want the control element to show (e.g. ”Internal” →”__System” →”Time”) and then click **OK**.

Specifying the VWItem property causes a change in what is shown in the ”VarOut1” control element. When the process variables to be shown are specified, the type of data is interpreted. The value shown in the control elements jumps to a value that simulates what is shown during the runtime. This makes it possible as early as the development time to check whether the control element can show the value completely or whether the dimensions of the control element have to be modified.

9. Select the **VarOut** control element and scale it until the clock time is shown.
10. Save the form by clicking the menu item **File** → **Save** and then look at the results by pressing **[F5]** to start the project.

Note!

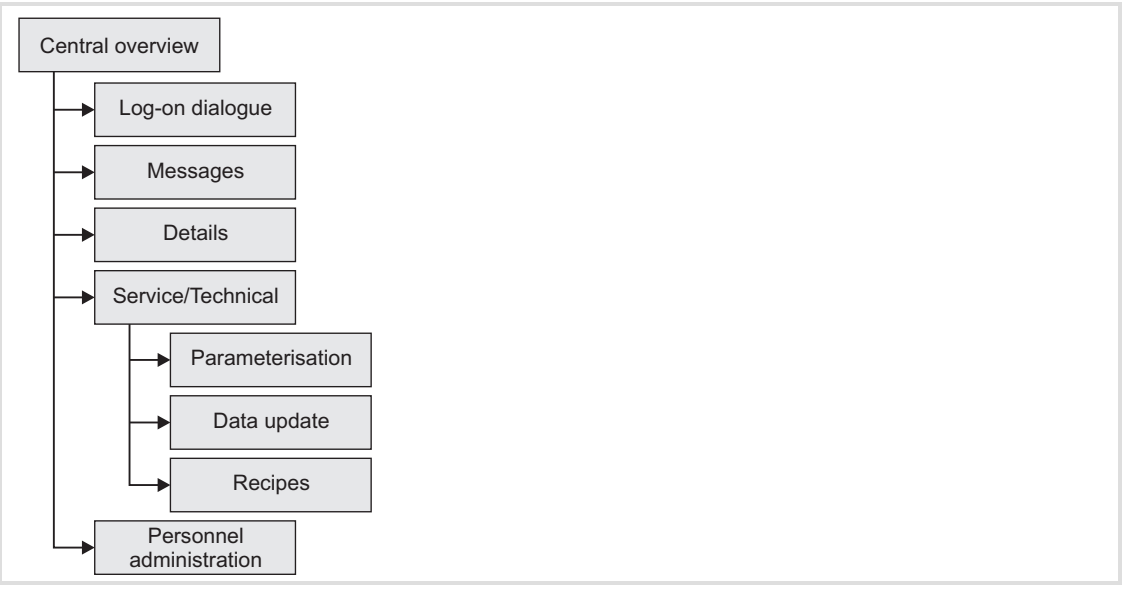
If you are working in demo mode (without dongle), an information dialog is displayed every time the runtime system is started. Close it with the **OK** command button.

4.4.2 Creating other forms and navigating between them

It is rare that a visualisation application will be able to manage with only one screen view. The usual and more useful approach is to divide it into different screen views according to the way in which the plant functions or according to its logic. Starting from a central overview, the user can make use of command buttons to change to the parts of the application that concern him and then make his entries and carry out his observations there. In this context, it is necessary to bear in mind that functions are enabled in relation to particular individuals. For example, it is usual that only a technician and not the operating personnel are allowed access to a Service page.

Basic structure of a visualisation application

A basic consideration when a visualisation application is being set up is the way in which the screen views are divided up. The following is a typical example:



Each of the function blocks shown here represents a screen view. The hierarchy shows how the forms are to be logically linked to each other, i.e. which form can be called and from where. If, for example, you are in the central overview and want to gain access to the individual function blocks underneath it, at least five command buttons are necessary.

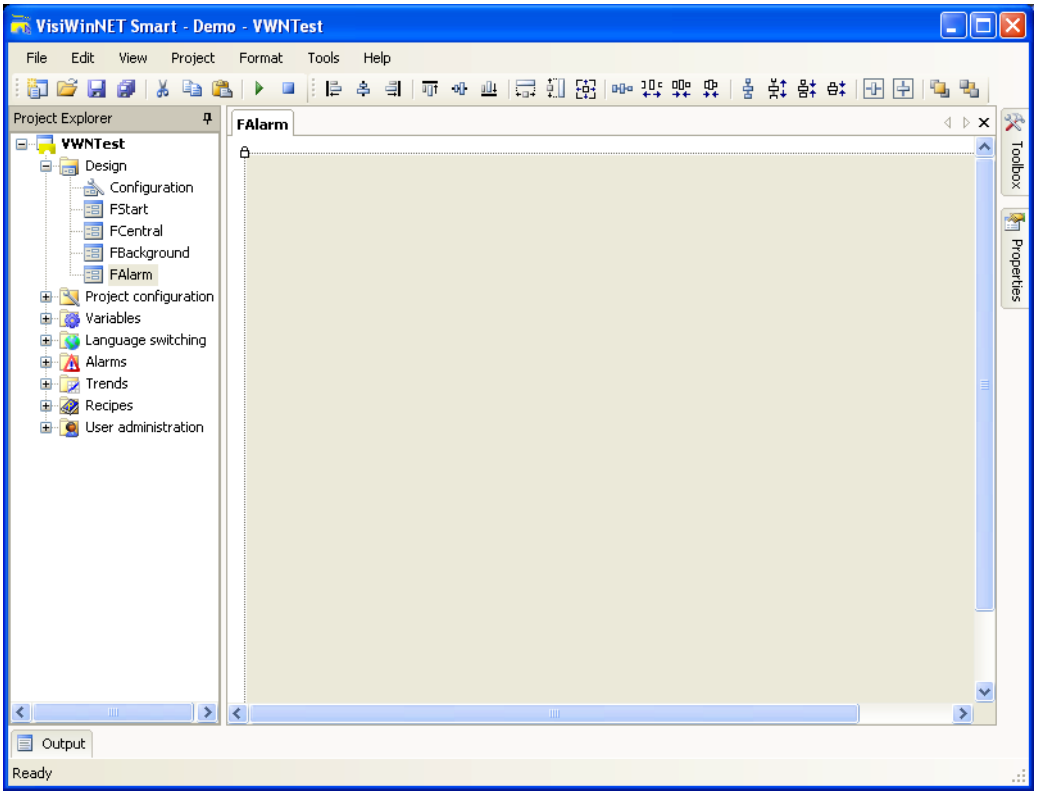
It is a question of taste whether screen pages that are parallel to each other need to be logically linked to each other. However, one exception is without doubt an alarm view page that it should always be possible to reach from any part of the application.

Creating a new form

1. Create a new form by right-clicking the **Design**node and activating the **Add Form** command in the context menu.

   A new object with the name ”Form” is inserted under the ”Design” node and an empty form is opened.

2. First of all, change the name of the form by right-clicking the ”Form” object, selecting the **Rename** command and replacing the old text with **FAlarm**.



3. Open the Properties of the form by right-clicking the ”Form” object and then select the **Properties** command.

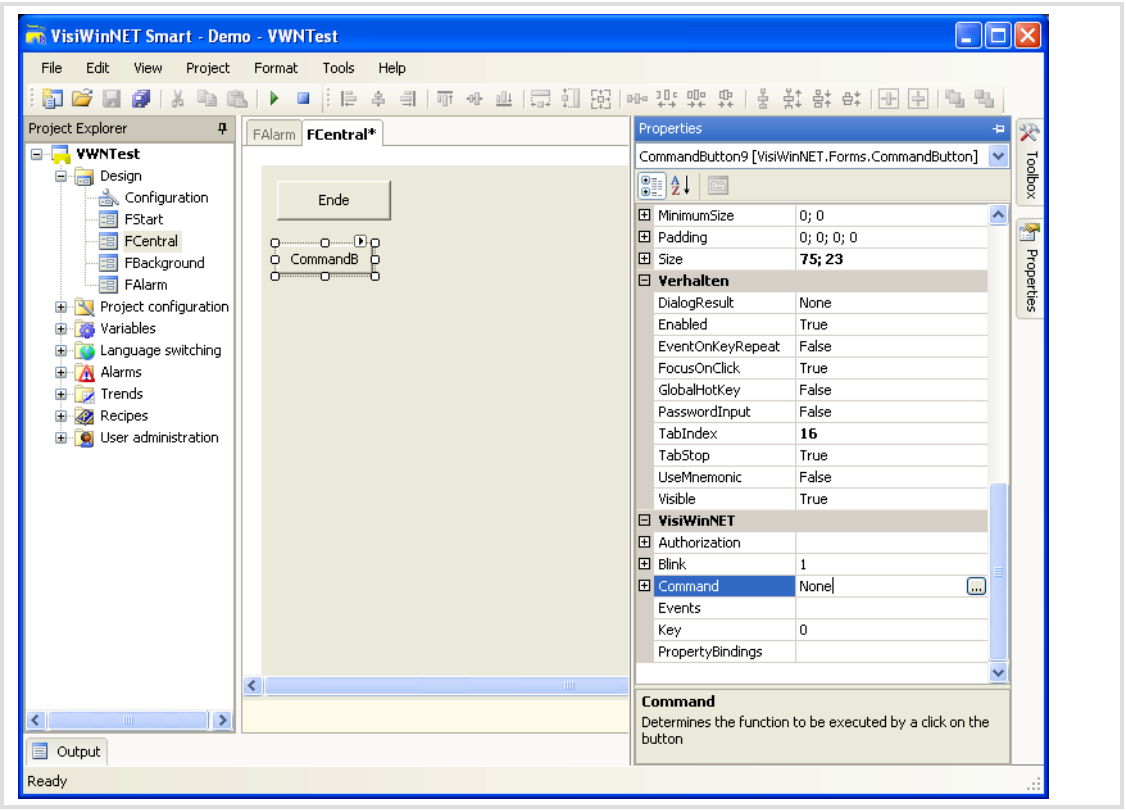
   The Properties window then opens. Here, you can configure the following properties:

   – ExecuteCode
   – LoadOnStartup
   – VariableActivationMode
   – VariableDeactivationMode

Creating a command button for navigating between forms

1. Use the tabs to change to the **FCentral** form.
2. Click the tab on top right-hand side of the **Toolbox** in order to show the Toolbox.
3. Under the **Common** Toolbox section, select the **CommandButton** type of control element.
4. In the form, click the place where the control element is to be inserted.
5. Select the new control element and click on the **Properties** on the top right-hand side in order to display the Properties window.

   The Properties window shows the current parameters of the control element.

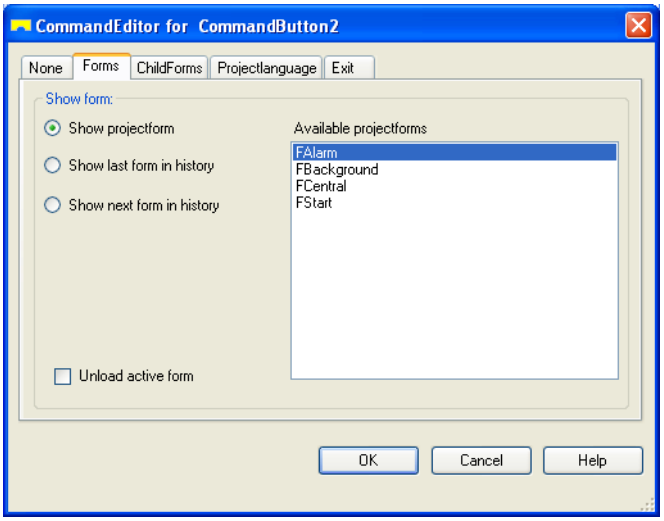


   The type of control element chosen is used for the execution of commands. This is defined by means of the ”Command” property.

6. In the **VisiWinNET** properties section, mark the **Command** property and click on the **[...]** command button there.

   The dialog ”Command’Editor ...”shows the available commands, listed in order in the tab.

7. Activate the **Forms** tab.



8. Select the option field **Show project forms**and, in the selection field, **Available project forms**, select the **FAlarm** form. Close the dialog with **OK**.
9. Now, change the labelling of the command button by going to the **Text** property and entering **FAlarm**.

   Before you test the project: You have now defined a command button by means of which you can jump from the ”FCentral” form to the ”FAlarm” form during the runtime. However, what is still missing is a way of jumping back.

Note!

If you do not create a way of jumping back, you cannot exit the runtime system. The only resort is to exit VisiWinNET®Smart by means of the Windows task manager.

You can now position a command button on the ”FAlarm” form, analogously to the described procedure, whereby this command button enables you to jump back to the ”FCentral” form. However, we would like to show you an alternative.

10. Change to the **FAlarm** form, select the form and open the Properties window.
11. Under **FormBorderStyle**, choose the entry **FixedDialog**.

    You have thus assigned the appearance and function of a standard Windows window to the form on the top right-hand side with the three well-known command buttons for minimising, scaling and closing windows.
12. Save the form by clicking the menu item **File** → **Save** and then look at the results by pressing **[F5]** to start the project.

4.5 User administration

The VisiWinNET®Smart administration system is used to assign person-specific rights within the visualisation application. Input control elements can be deactivated and access to information via output control elements can be refused. VisiWinNET®Smart supports two administration systems for users:

Level-orientated user acknowledgement: Users and control elements can be assigned to one of 999 levels. If the level of a control element is higher than that of the recently logged-on user during the runtime, use of the control element is refused.

Right-orientated user administration: A right is assigned to the control elements in the application and the user must possess this right in order to be able to make use of a control element (e.g. ”Edit recipe”). The rights are then assigned to the users who are permitted to use the control elements. This is useful, especially in a client-server system, if several employee hierarchies (groups of employees who have been divided into authorisation levels independently of each other) work in the network. The decision as to whether users are to be enabled to use the control elements is made on the basis of a comparison of the rights of users and control elements.

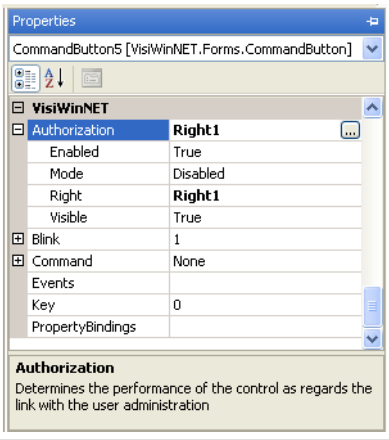
User administration is carried out on the basis of three definitions in the project database:

User group: contains a group of specific users and the characteristic parameters for all the users contained in it. The user group is used to manage the references to the rights needed to make use of the application.

User: contains the access information (log-on name and password) and the activation status of a user.

Right: Rights determine the information assigned to the control elements of the application. The ”Authorisation” property is used to assign a right to a control element. A logged-on user is permitted to make use of the control element only if the user-group definition of the user contains a reference to this right. Rights are only used in the right-orientated form of user administration. In the level-orientated form of user administration, the ”Authorisation” property of the control element expects a number.

All VisiWinNET® Smart control elements contain an ”Authorisation” property. In this property, the name of the right that enables a control element to be used during the runtime has to be specified. It is also necessary to specify how the control element behaves in the event of a lack of authorisation. If the property is expanded in the Properties window, it is possible to choose between ”Disabled” and ”Invisible” in the ”Mode” setting field.



During the runtime, VisiWinNET® Smart provides special dialogues for the management of user information. A typical example is the Log-on dialog, which asks the user to enter the user name and a password. After log-on, the control elements are automatically activated or disabled, depending on the authorisation assigned to the user in question. Calling of the user-administration dialogues is supported by the component ”UserManagementDialogs”.

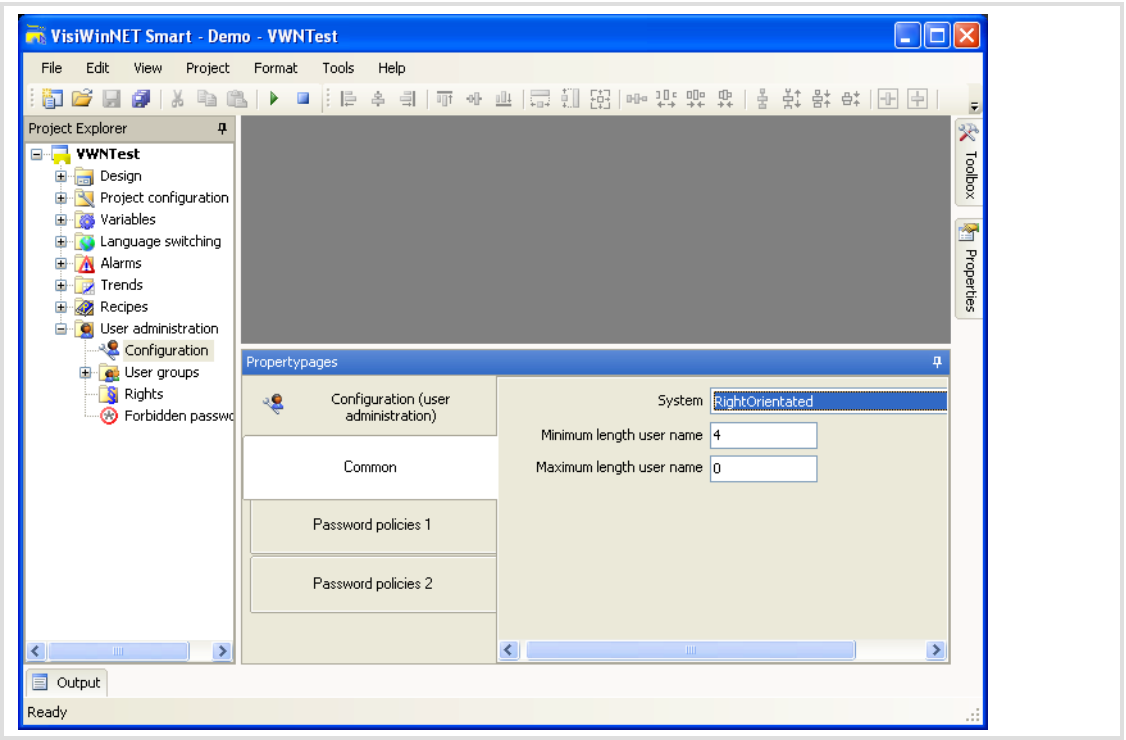
The following example shows the steps involved in implementation of a right-orientated form of user administration. How the individual parts have to be logically linked and how user data can be administration during the runtime is shown in the example. The example includes an administrator and two users with different rights.

With the help of the ”Authorisation” property, the corresponding enabling functions are logically linked in the control elements.

The example also shows how to open the Log-on dialog and the Administration dialog for updating user data.

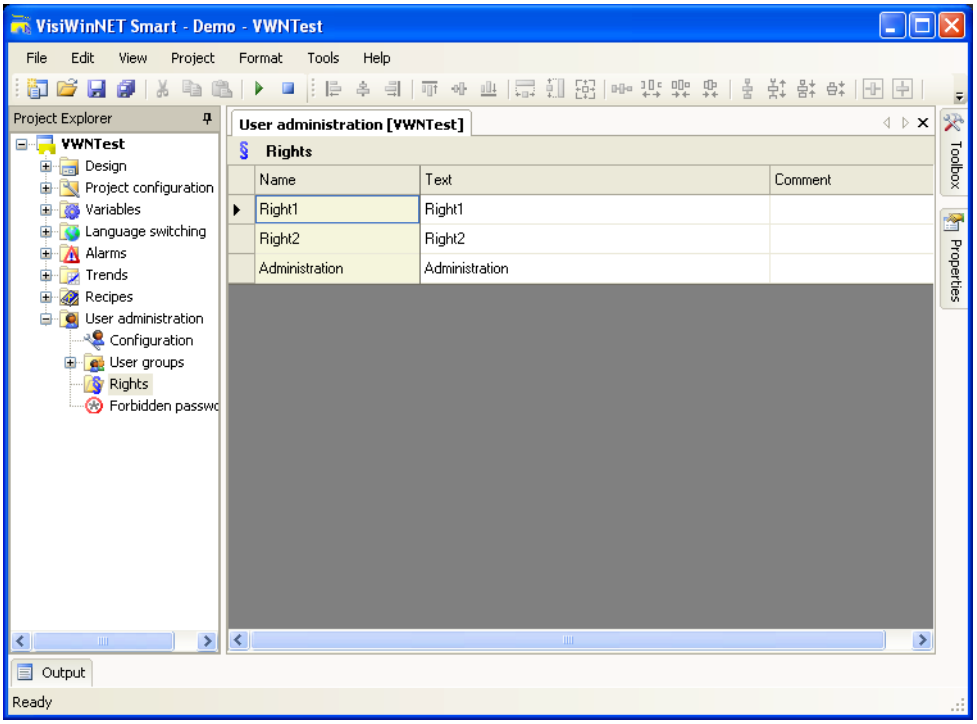
4.5.1 Defining the type of user administration system

1. In Project Explorer, select the node **User administration** → **Configuration** and, on the Properties page, under **System**, specify the value **Right-orientated**.
   In Project Explorer, the ”Rights” node is then shown.



4.5.2 Creating rights

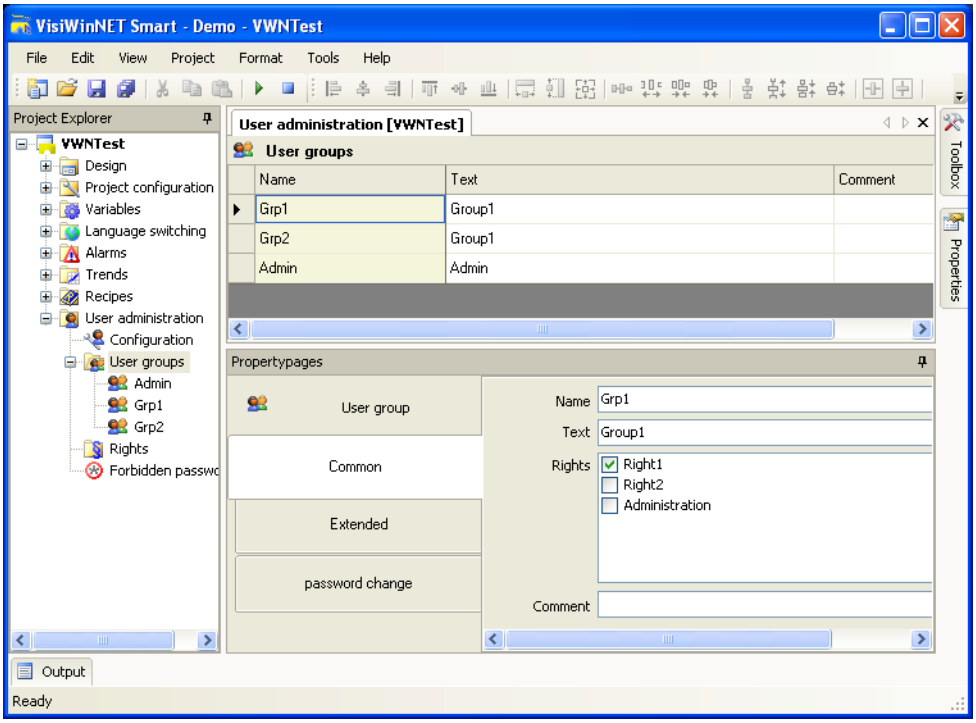
1. Double-click the **Rights** node.
2. Right-click in the client area and select the **New** command in the context menu.
   A new right (”Right1”) is generated in the table editor.
3. Create two more rights in the same way (”Right2” and ”Administration”).



4.5.3 Creating user groups

Rights are assigned to users by means of the user group. At first glance, this seems to be somewhat complicated but is helpful in the real world. Each user group can accommodate any number of users. The rights for all the users in the group can be set at the same time, i.e. they can be adapted simultaneously.

1. In Project Explorer, select the **User groups** node.
2. Create a user group. To do this, right-click in the client area and, in the context menu, select the **New** command.



3. Create two further user groups.

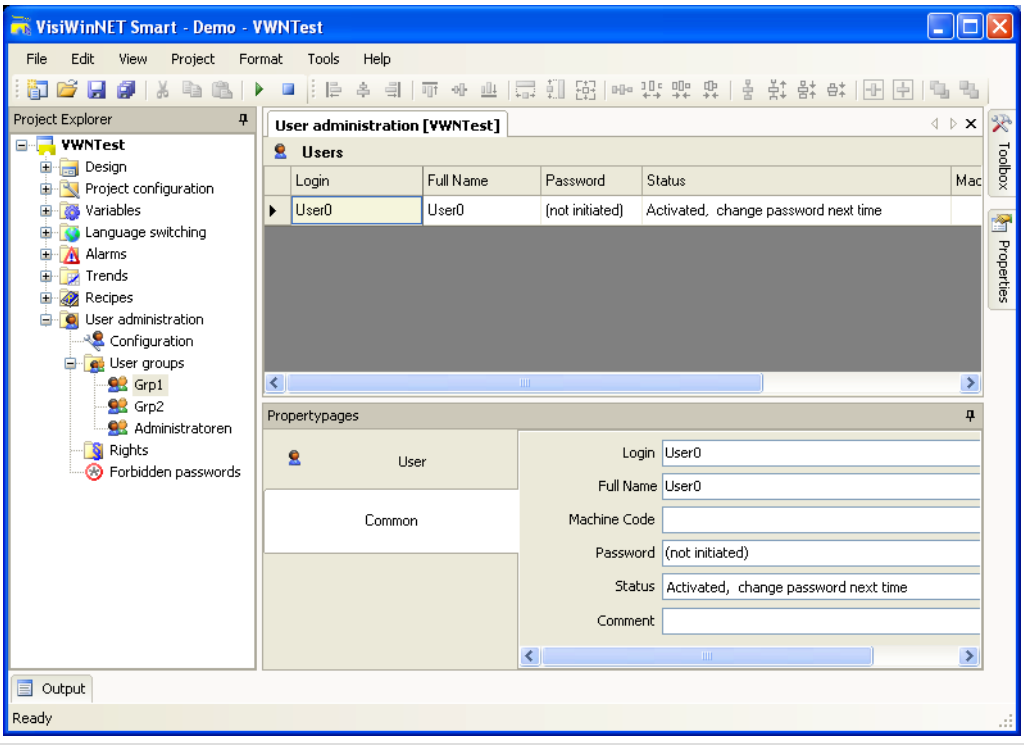
4. Assign the following properties to the three user groups:

Properties for user groups		
No.	**Tab/Property**	**Value**
1	General	
	Name	Grp1
	Text	Grp1
	Right	Right1
2	General	
	Name	Grp2
	Text	Grp2
	Right	Right2
3	General	
	Name	Administrators
	Text	Administrators
	Right	Right1, Right2, Administration

Administrators are allowed to do more; everything in this case.

4.5.4 Creating users

1. In Project Explorer, open the **User groups**node and click the **Grp1** element.
2. Right-click in the client area and select the **New** command in the context menu.
   In the table editor, a new user is then created (”User0”). This user is a member of user group ”Grp1”.



3. Create a user in the ”Grp1” user group and ”Administrators” as well.

4. On the respective Property page, assign the following properties to the users:

Properties for users		
Group	**Tab/Property**	**Value**
Grp1	General	
	Name	Max
	Full name	John Smith
	Password	0000
	Status	activated
Grp2	General	
	Name	Fred
	Full name	Fred Smith
	Password	0000
	Status	activated
Albert.	General	
	Name	Admin
	Full name	Albert Smith
	Password	0000
	Status	activated

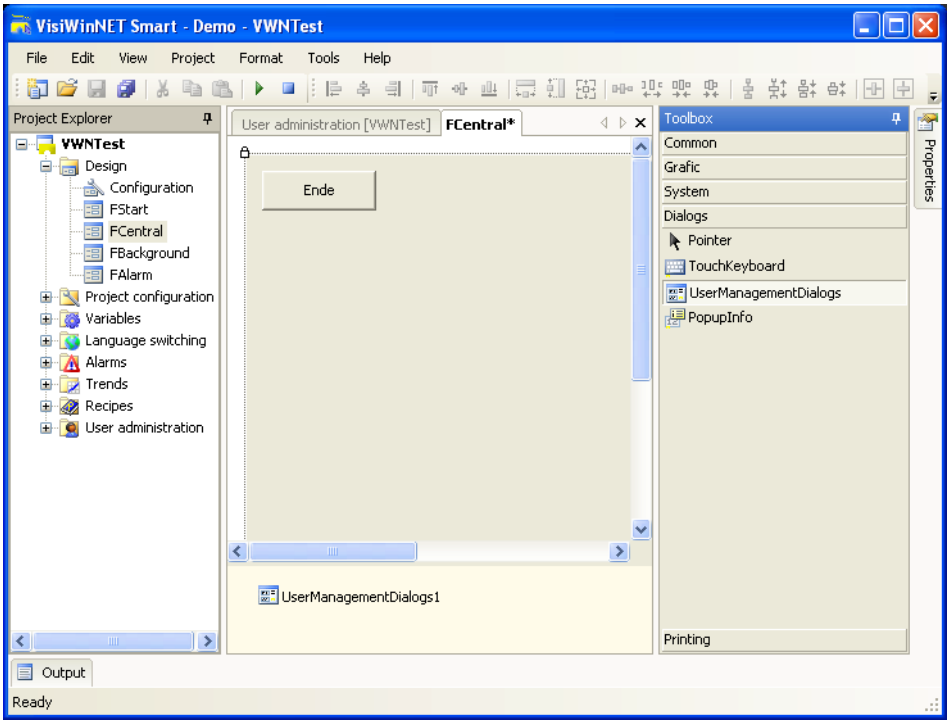
Log-on name and password are used later as log-on information. The status enables different settings that decide whether the user is permitted to log on and whether he has to change the password assigned by the project engineer the first time he logs in.

4.5.5 Using dialogues for user administration

1. Open the **FCentral** formand place a toolbox control element of the type ”UserManagementDialogs” (”Dialogs” section) in it.

   In this way, you provide the project with diverse standard dialogues for user administration.

   After the control element has been placed on the form, it automatically ’slides’ into the so-called ”component slot” below the form.



2. Place four control elements of the type ”CommandButton” on the **FCentral** form:

Properties for control elements			
No.	**Control element type/Property**	**Value**	**Comments**
1	CommandButton		
	Display/Text	LogOn	
	VisiWinNET/Authorisation		Empty = accessible for all users
	VisiWinNET/Events	Logically links click event with calling of the **ShowLogOnDialog** function of the **UserManagementDialogs1** component.	If the **[...]** command button is clicked, a dialog opens. Here, the click event must be selected and the **Select** command button must be clicked.
2	CommandButton		
	Display/Text	Grp1	
	VisiWinNET/Authorisation	Right1	Only accessible for users of the user group with Right 1 (=Grp1)
3	CommandButton		
	Display/Text	Grp2	
	VisiWinNET/Authorisation	Right2	
4	CommandButton		
	Display/Text	Admin	
	VisiWinNET/Authorisation	Administration	
4	VisiWinNET/Events	Logically links click event with calling of the **ShowUsersDialog** function of the **UserManagementDialogs1** component.	

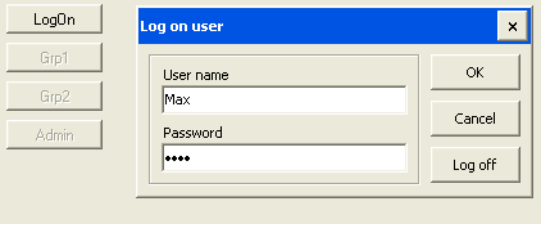
The purpose of the text property here is to show the CommandButton function. The "Authorisation" property specifies when the command button is to be activated. If a user has logged on and if his user group is logically linked to the right indicated here, the command button is activated.

The purpose of the "Events" property setting is to call the "Log-on user" and "Administration" dialogues.

3. Start the project with **[F5]**.

   The "LogOn" command button has been activated. No setting has been entered in the "Authorisation" property.

4. Click the **LogOn** command button.



The dialog for user log-on is opened. The three command buttons "Grp1", "Grp2" and "Administration" are activated by log-on of the corresponding user.

5. Try out your user names with your passwords.

   The two command buttons ”Grp1” and ”Grp2” are inactive. They are activated by the users from the groups ”Grp1” & ”Administration” and ”Grp2” & ”Administration” respectively. If command buttons that have been parameterised in this way are used to switch over to other forms, entire parts of the application can be unlocked, depending on the current user.

   The ”Administration” command button is only available to users from the user group with the same name. Clicking this command button causes the user administration dialog to open. Here, new users can be created and existing ones can be managed.

4.6 Creating multiple-language projects (language switching)

The language switching function makes it possible to change all the texts used in the application according to the language selected. For example, it is used to ...

- deliver an international application. In this case, it would be enough to configure the language to be displayed when the project is started or to be displayed on a service page.
- enable operation by international operating personnel. For this purpose, a function could be provided that would switch over the language when there is a change of shift involving a change in the nationality of the personnel.

Both are possible at the push of a button.

Apart from the actual texts that are used in the application, other special aspects must be taken into consideration:

- In accordance with the language selected,
  – units (e.g. temperature °C/°F) have to be converted and adapted
  – date and time formats have to be shown in the local format.
- It is possible that the size of fonts in different character sets will be scaled differently. This means that the font will have to be adapted when the language is changed.

Further information on the subject ”Development of multilingual visualisation applications” can be found in the appendix (📖 261).

4.6.1 Calling the language switching function

1. Expand the **Language switching** node in Project Explorer.

   The following subnodes are shown:

   The **Configuration** node contains the following subgroups:

   – **Font classes** and **Fonts** define how font switch-over functions.

   – **Languages** contains the project languages in the form of a list. A new project contains German and English. Further languages can be added to the project.

   User texts are the texts that the user can create in order to display them in the application.

   Components and **Dialogs** contain the texts that are used and expected by the control elements and dialogs of the VisiWinNET package. Texts cannot be created here; the texts can be supplemented with texts in other languages or can be altered in relation to the particular application in question.

   Texts cannot be created in the **Variables**, **Alarms**, **Archive**, **User management** or **Logging** subnodes. This is done by the editors of the corresponding systems. If, for example, an alarm is created in the alarm system, the text of the alarm (the language of which can be switched over) appears under the node that is displayed here. This is in line with the central approach to management of the language switching function.

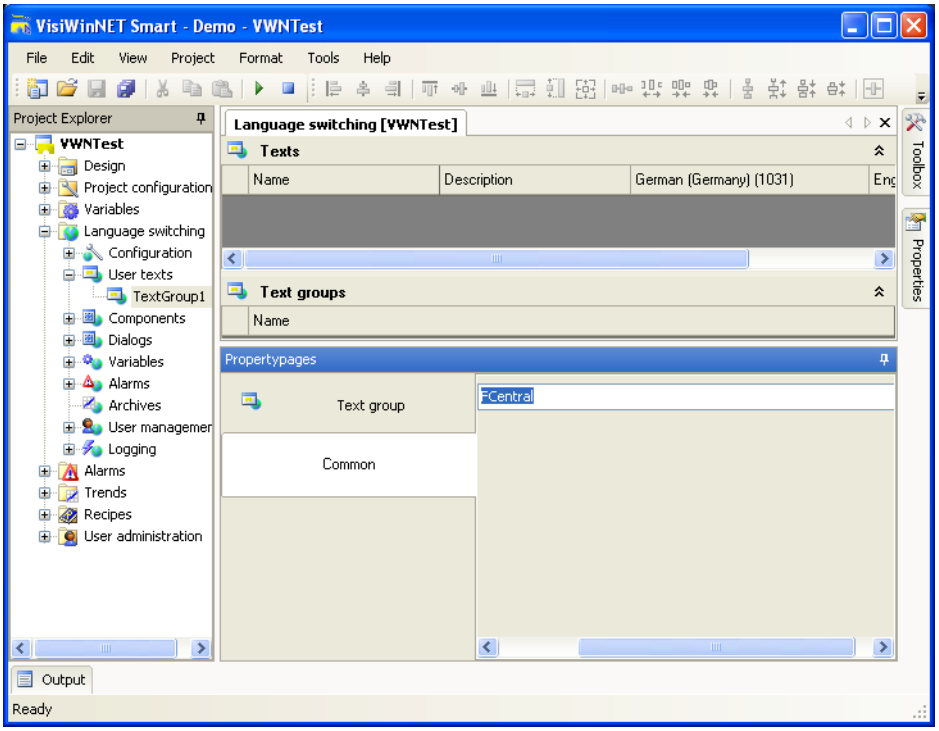
Note!

The unit switch-over function is a part of process integration. Conceptually, it is kept separate from the language switching function as the information to be defined contains conversion values as one of its main features.

4.6.2 Creating user texts

1. In Project Explorer, right-click the node **Language switching** → **User texts** and, in the context menu, select the **New** command.

   The subnode ”Text group1” is created below the User texts node. . In the Client area, the ”Texts” and ”Text group” object lists are shown.



2. First, change the name of the text group to **FCentral** on the Properties page.

   Texts whose language can be switched over can be created and managed in individual groups. This has the following advantages:

   – By means of the name of the text group, it is possible to specify where the texts will be used. Text groups can thus be used to create a structure.

   – The texts that are to be created and then perhaps altered sometime later are easier to find.

   – Creating a text group for each form makes later work easier. If the form is to be used in another project, only the corresponding text group has to be copied to the project. The form that uses the texts is then capable of functioning again.

3. Right-click the **Text** object list and, in the context menu, select the **New** command.

   A new line is created in the ”Text” object list.

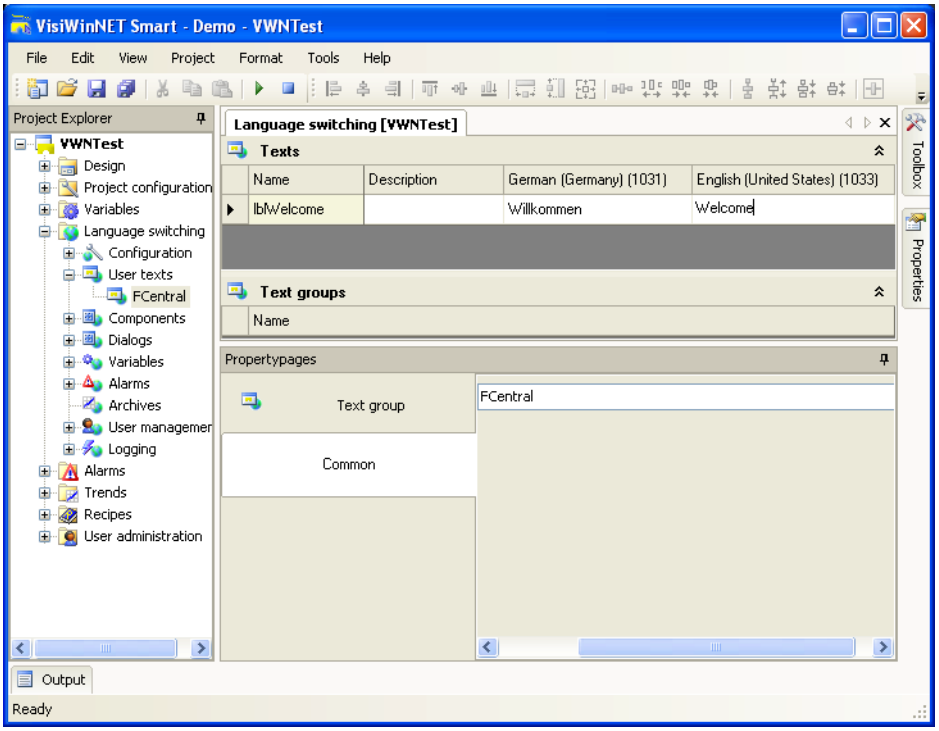
4. Enter the following texts in this line:

   Name = lblWelcome

   Description = *Any text or empty*

   German = Willkommen

   English = Welcome



You have now defined your first user text.

4.6.3 Assigning user texts to a control element of the type ”Label”

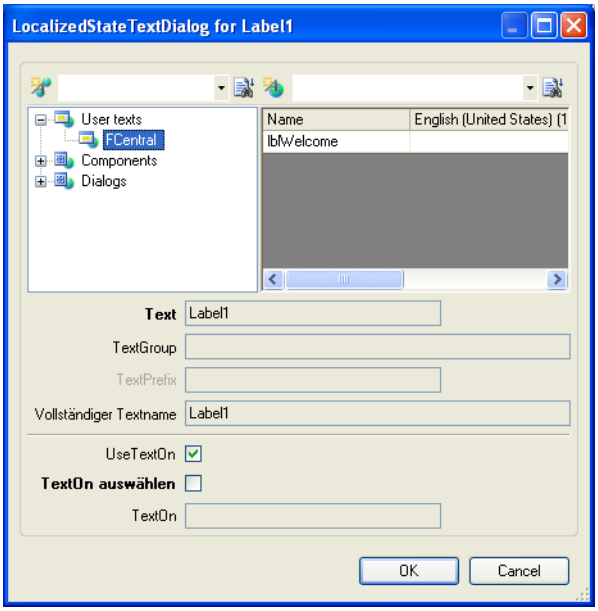
Next, we will create a control element of the type ”Label” on the ”FCentral” form and logically link it to the user text.

1. Expand the **Design** node in Project Explorer and double-click the **FCentral** form object.
2. Drag the **Label** control element from the Toolbox onto the **FCentral** form.
   The control element will then be shown on the form and the ”Label task” tasks menu will be opened.

Note!

The tasks menu is closed as soon as you left-click somewhere outside the menu. You can open it again by marking the control element and clicking the small button ▶at the upper edge of the control element.

3. While in the **Label task** tasks menu, click **LocalizedText (Label1)**.
   The dialog entitled ”LocalizeStateTextDialog for Label1” will be opened.
4. In the **Text group** selection field, open the **User texts** node and double-click the **FCentral** text group object that we created previously.
   The user text ”lblWelcome” is shown in the list field.



5. Highlight the text **lblWelcome** in the list field and click **OK**.

   You have now defined the text that is to be shown by the control element during runtime. After acceptance of the text is confirmed, the latter is shown in the control element in the active language.

Note!

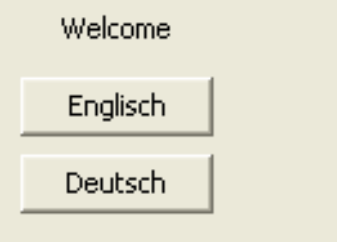
Apart from the label control element, other control elements also directly or indirectly support the display of texts whose language can be switched over. In general, the following applies: a ”LocalizedText” property indicates that a text whose language can be switched over is displayed.

Other control elements have no such property but they do obtain texts or formatting instructions from the language switching function. The ”AlarmLine” control element, for example, includes the ”DateTimeFormat”’ property, which enables the specification of a text from the language switching function. The indication of ”@LongDate” as the value of a property says that the text of the same name from the ”Components.Time.DateFormats” text group contains a formatting character string for each project language, whereby this string is used to format date/time values during the runtime. This is also done by the language switching function.

4.6.4 Switching over the language during the runtime

Now, we want to assign two buttons to the ”FCentral” form, whereby the two buttons can be used to switch over between languages during the runtime.

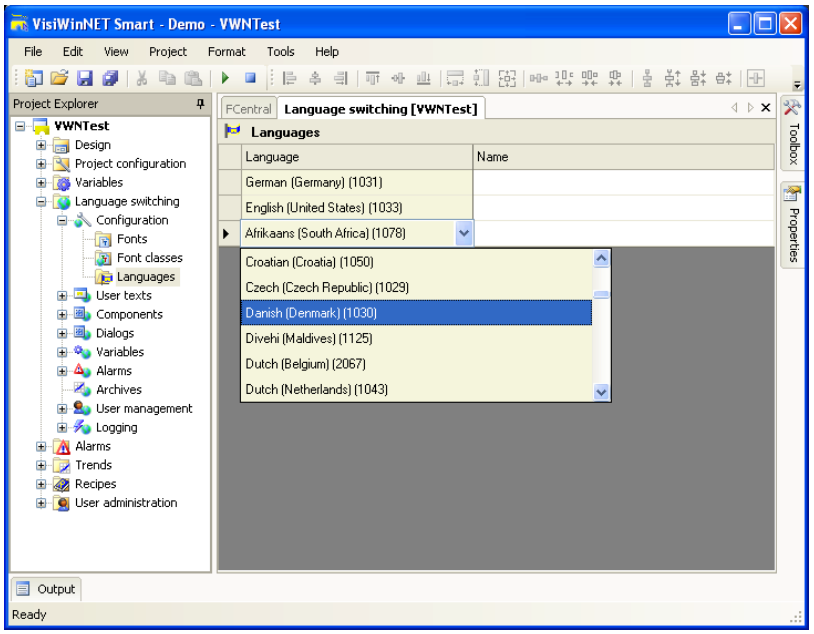
1. Pull the **CommandButton** control element from the Toolbox onto the **”FCentral”** form.
   The control element will then be shown on the form and the ”CommandButton Task” menu will be opened.
2. Close the **CommandButton Task** menu by clicking anywhere on the form.
3. Highlight the new control element and open the Properties window.
4. In the Properties window, change the name of the control element:
   View → **Text:** English
5. In the Properties window under **VisiWinNET** → **Command**, click **[...]**.
   The ”CommandEditor for ...” dialog opens.
6. Activate the **Project language** tab and, under **Available project languages**, select the entry **English (United States)**.
7. Click **OK** to close the dialog.
   In the Properties window under **VisiWinNET** → **Command**, the entry ”ChangeLanguage(1033)” now appears. You have thus assigned the command to switch-over the user texts to ”English” to this button.(1033 is the ”Local Identifier” for American English; 1031 stands for German.)
8. Repeat the steps to create a further button which you can use to switch back to German as the user language.



9. Press **[F5]** to start the project and test the various functions.

4.6.5 Defining further languages

1. In order to make other languages available, double-click the node **Language switching** → **Configuration** → **Language** in Project Explorer.
2. Right-click in the client area and select the **New** command in the context menu.
   A new line for a new language is created.
3. Open the selection field in the new line and select the desired language.



Note!

The specified regional and language settings are not available?

Regional and language options are provided by the operating system to help the user. They change the country-specific characters/special characters on the keyboard.

A German keyboard, for example, has the umlauts ä, ö and ü, which do not exist in English. On English keyboards, other characters are assigned to these keys. The key code that is transmitted by the keys, however, is identical.

The operating system controls the assignment of keys to the characters printed on the keys by means of a country-specific table, the ”Regional and language settings”.

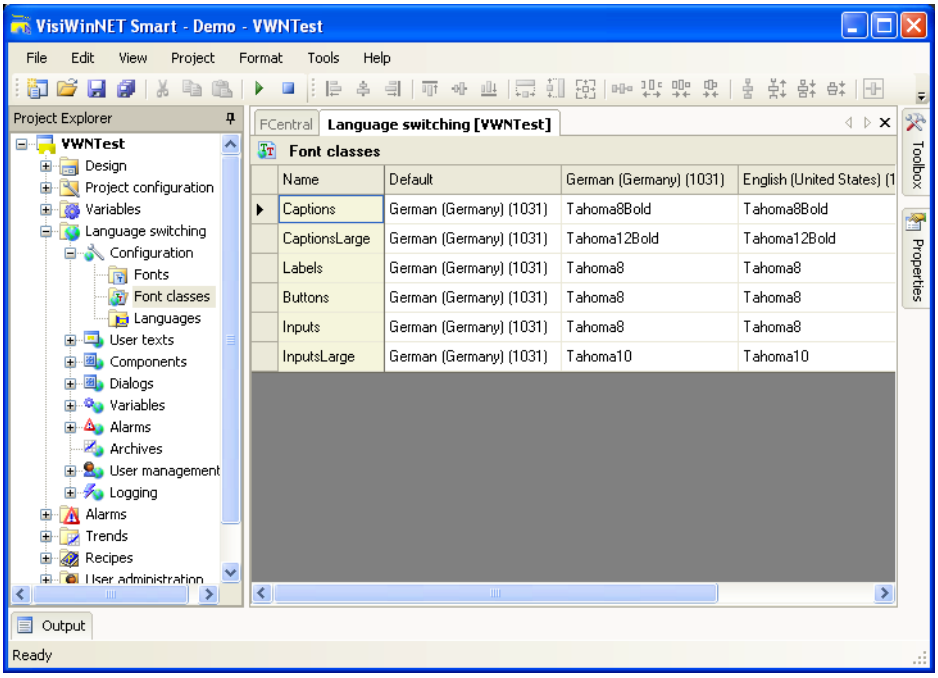
The different regional and language settings that are needed have to be installed: Windows Start menu →Settings→Regional and language options →”Languages” tab → ”Details” button).

Here, you must enter the desired language. The regional and language settings for this language are then shown in the selection list of the table editor field.

If a special font is needed for texts to be shown in Danish, for example, this can be specified with the help of the ”Fonts” and ”Font classes” definitions.

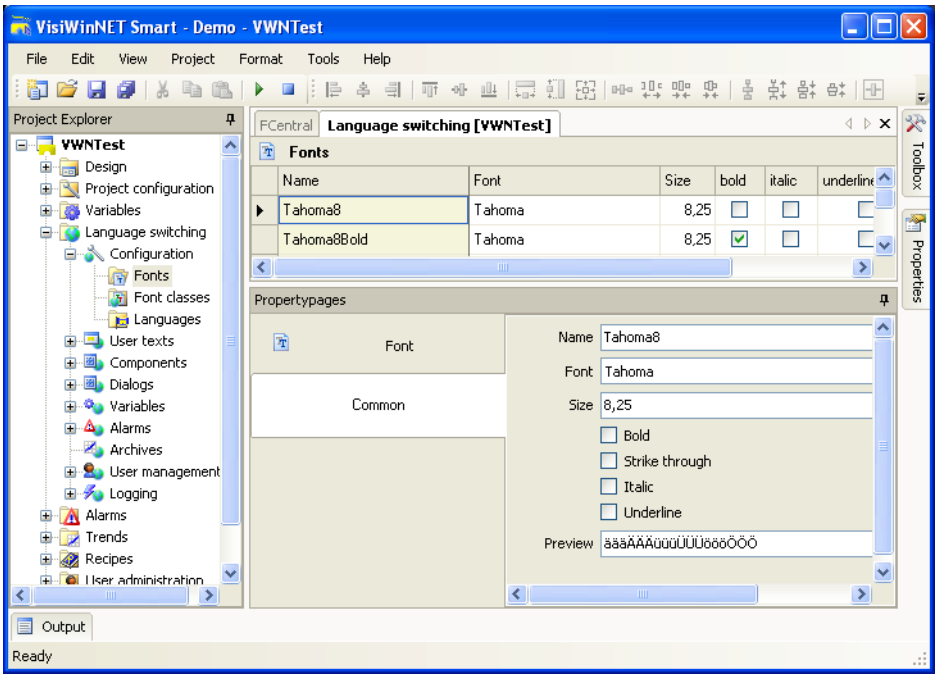
4. To do this, double-click the node **Language switching** → **Configuration** → **Font classes** in Project Explorer.

   The ”Font classes object list” opens in which you can specify font classes for each language.



You can specify the font classes themselves by means of **Language switching** → **Configuration** → **Fonts**.

The font class according to which the text of a control element is to be shown is assigned by means of ”FontClass” in the associated Properties window.

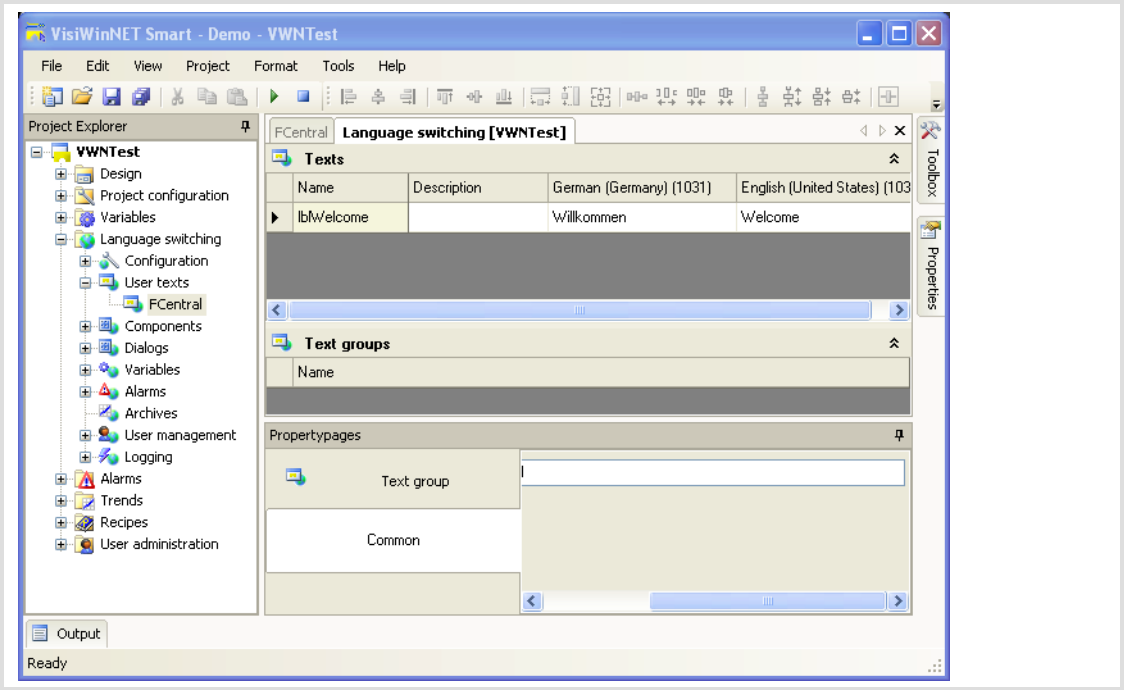


4.6.6 Using dynamic texts (showing variables)

In this section, we describe how you can arrange for information generated during the runtime to be displayed. Such dynamic items of information are, for example, process variable values or date/time formats.

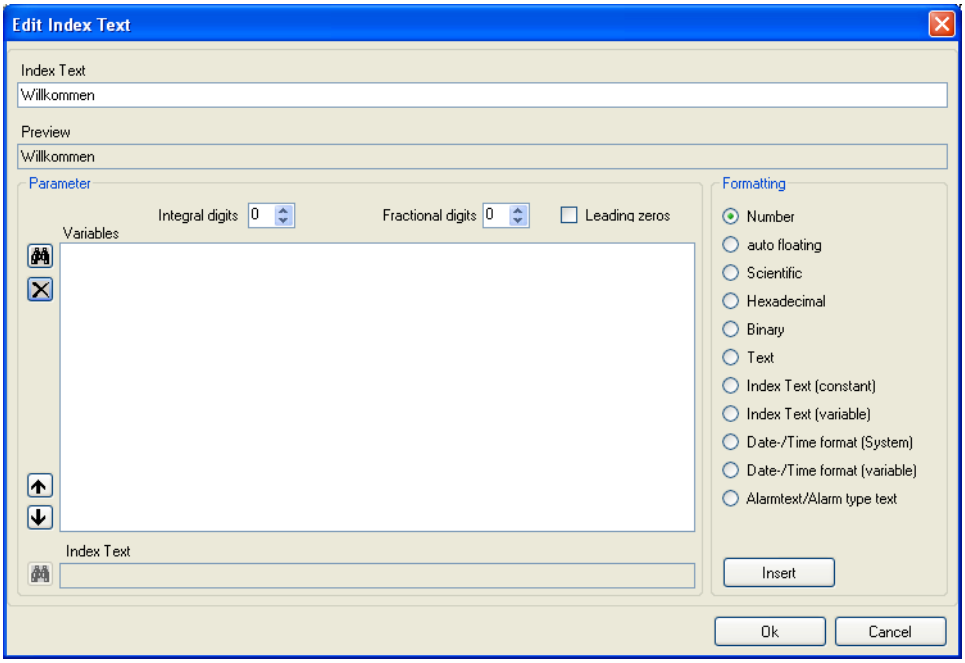
During the development time, format character strings enclosed by ”@” characters are inserted into the text. These are replaced with the corresponding values during the runtime.

1. Double-click the node **Language switching** → **User texts** → **FCentral**
2. in Project Explorer.
   In the client area, the ”Texts” and ”Text group” object lists are then shown.

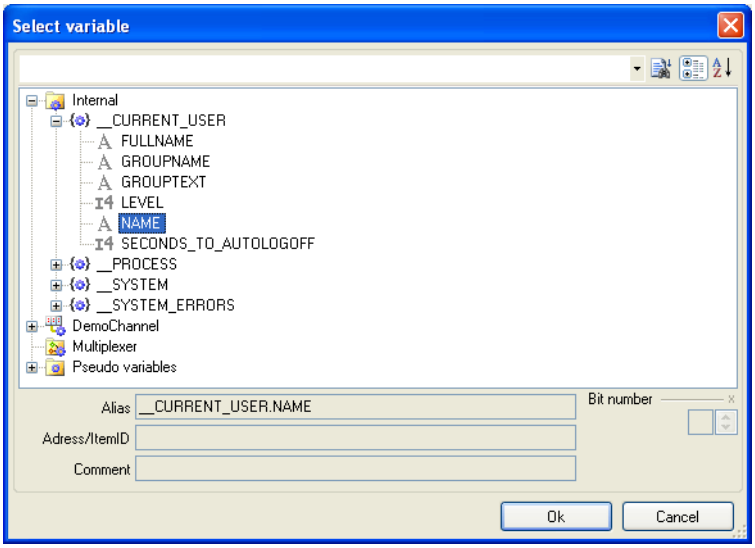


   You have now defined your first user text.

3. Highlight the field in the line **lblWelcome** and the column **German (Germany)**.
4. Click the **[...]** button.
   The ”Edit index text” dialog appears.



5. Click the button with the binoculars to select a process variable.



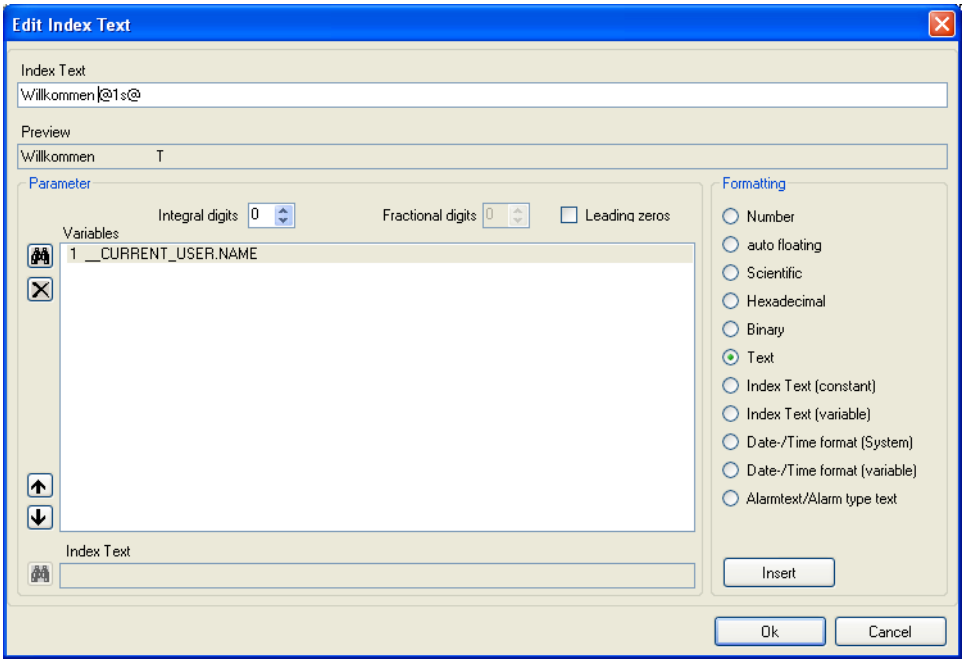
6. In the **Select variable** dialog under the node **Internal** → **__CURRENT_USER**, select the variable **NAME**and close the dialog with **OK**.

   The variable ”__CURRENT_USER” is transferred to the selection field in the ”Edit index” dialog.

7. Highlight the variable and the **Text** formatting option field. Then, click **Insert**.

   In the ”Indextext” text field, it now says: ”Willkommen@1s@”

   The formatting character string enclosed by ”@” characters indicates that the first process value (1) in the process variables list is to be output as text (s) at this point.



8. Insert a space between ”Willkommen” and ”@1s@” and close the dialog with **OK** .
9. Now, highlight the field in the column **English (United States)** and click the **[...]** button.

   The ”Edit index text” dialog appears again. The variable **__CURRENT_USER. Fullname**has already been pre-assigned to the list field.
10. Highlight this variable and the formatting option field **Text**. Then, click **Insert**.

    In the ”Indextext” field, it now says: ”Welcome @1s@”
11. Insert a space between ”Welcome” and ”@1s@” and close the dialog with **OK** .
12. Start the project with **[F5]**.
13. Click the **LogOn** button and register as the user (📖 190).
14. Now, click the **German**and **English** buttons alternately.

German	English

4.7 Configuring the alarm system

The purpose of an alarm system is to centrally detect and store error states or specific alarms. The corresponding definitions of which variable triggers an alarm, how the alarm is displayed and what additional information belongs to the alarm have to be specified in the alarm editor. Notification regarding the variables that trigger the alarms is sent by VisiWinNET® Smart to the variables core for monitoring. If a change in the value of a variable occurs, the corresponding alarm is transferred to the application, where it is displayed in special control elements.

VisiWinNET® Smart contains two control elements that allow the direct display of active alarms:

AlarmList: All active alarms are displayed in the form of a list. The control element also supports the acknowledgement of alarms.

AlarmLine: The space-saving variant that allows the user to see active alarms in the area at the top or bottom of all forms.

In addition to the transfer of alarm data to the user interface, VisiWinNET® Smart allows alarms to be stored in files. This makes it possible to review the production history of a plant. The ”HistoricalAlarmList” control component is used to display these recorded data in the form of a list.

Alteration of a bit from the variables core is the means by which an alarm is controlled. It is possible to choose whether the bit is set to ”1” or ”0” so that an alarm is triggered. However, it is also possible to use an extension in order to check analogue variables to find out whether they exceed certain limits. Alarms can be acknowledged by the user. An acknowledgement feedback signal to the control system can also be implemented.

The appearance of alarms in the ”AlarmLine” and ”AlarmList” control elements is parameterised by means of alarm classes. Alarm classes contain different parameters (colours, status texts and symbols) that describe how an alarm is displayed. Each alarm makes reference to an alarm class. The display parameters of an alarm class are thus available to each alarm.

Alarm groups organise alarms into groups. In a way similar to language switching, it is useful to group the individual functional elements of a plant. This promotes clarity and endows the project with a modular structure. However, alarm groups also contain additional parameters that permit group-wise deactivation or acknowledgement of alarms.

The following example describes ...

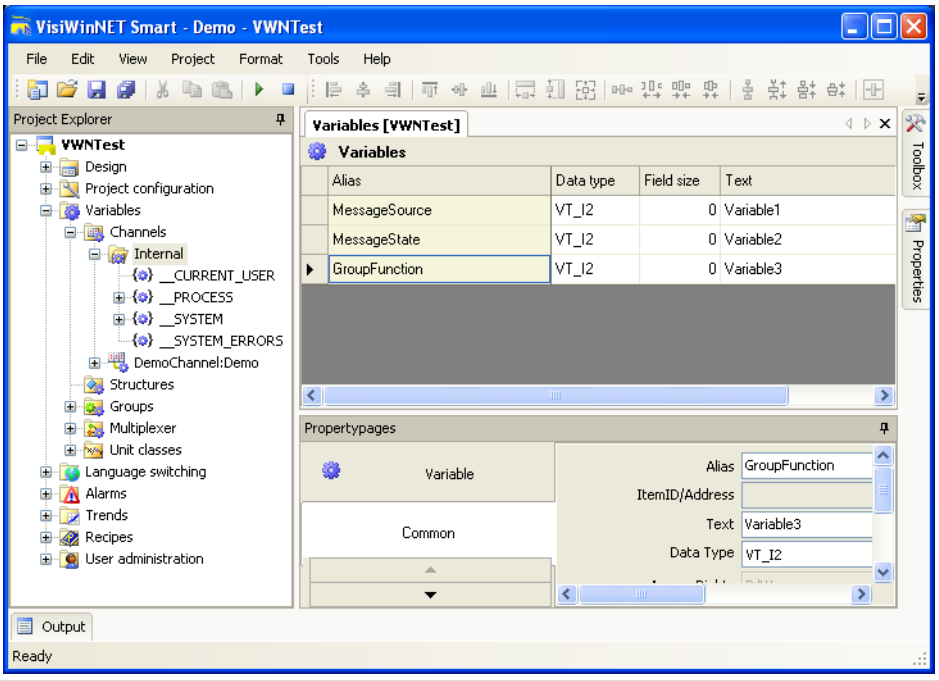
- how alarms that exist as bits in the variables core are generally configured, triggered and displayed
- the acknowledgement of alarms.
- various display filters.
- how analogue values are monitored.

First of all, the sources of the alarms have to be defined. Everything therefore begins during process integration due to the fact that alarms are tied to specific variables. However, because no real control data are available in the example, the triggering variables have to be simulated by means of internal variables.

4.7.1 Defining process variables

1. Double-click the **Variables** → **Channels** → **Internal** node in Project Explorer.
   The table editor for defining variables is opened.
2. Press the **[F8]** key to define the following three variables:

Properties for process variables		
No.	**Tab/Property**	**Value**
1	General	
	Alias	MessageSource
	Data type	VT_I2
2	General	
	Alias	MessageState
	Data type	VT_I2
3	General	
	Alias	GroupFunction
	Data type	VT_I2



The variables defined here are sufficient as a functional data link:·

– **MessageSource** contains the bits that trigger the alarms.
– **MessageStates** contains the alarm states sent back from the core functions.
– **GroupFunction** is used to control alarm functions by means of the higher-level group.

The definitions are now applied in the alarm system.

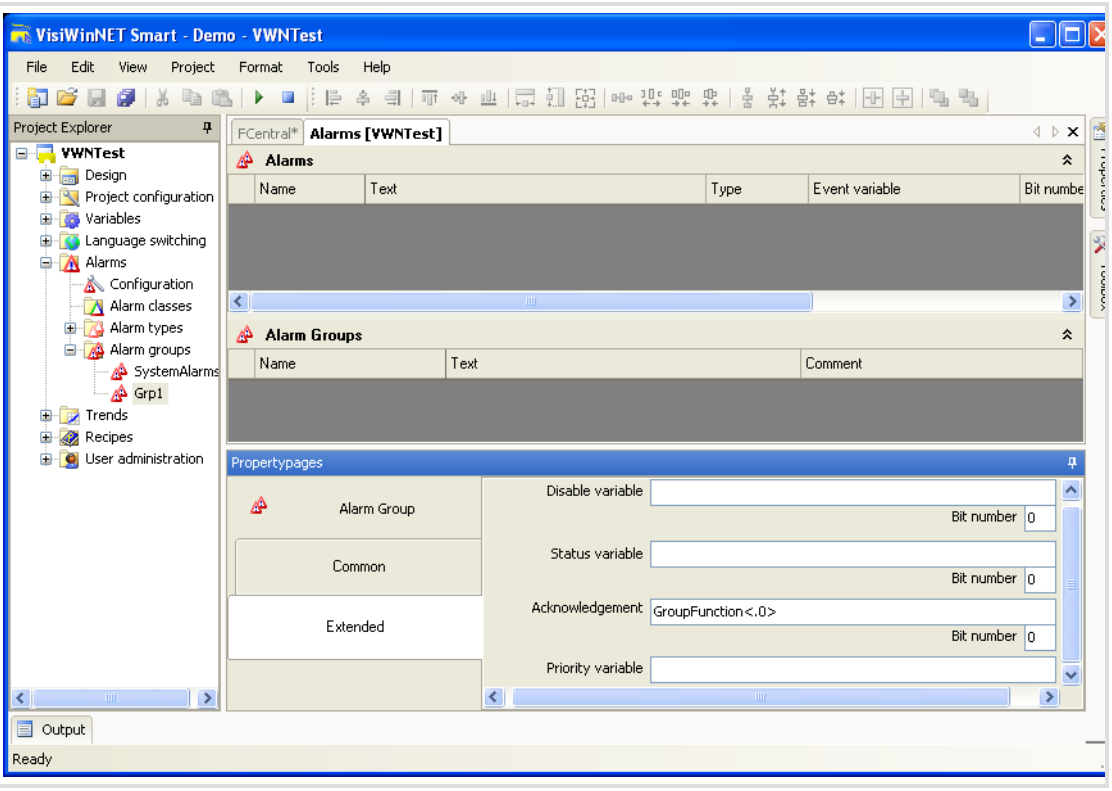
4.7.2 Creating alarm groups

1. In Project Explorer, right-click the node **Alarms** → **Alarm groups** and, in the context menu, select the **New** command.

   The table editor for defining alarms and alarm groups is opened.

2. Specify the following properties:

Properties for alarm groups		
No.	**Tab/Property**	**Value**
1	General	
	Name	Grp1
	Acknowledgement variable	GroupFunction
	Bit No. of acknowledgement variable	0



3. In Project Explorer, right-click the node **Alarms** → **Alarm groups** and, in the context menu, select the **New** command.

   Alarm groups can be nested. You must therefore first change to the ”Alarm groups” node!

4. Specify the following properties:

Properties for alarm groups		
No.	**Tab/Property**	**Value**
1	General	
	Name	Grp2
	Acknowledgement variable	GroupFunction
	Bit No. of acknowledgement variable	1

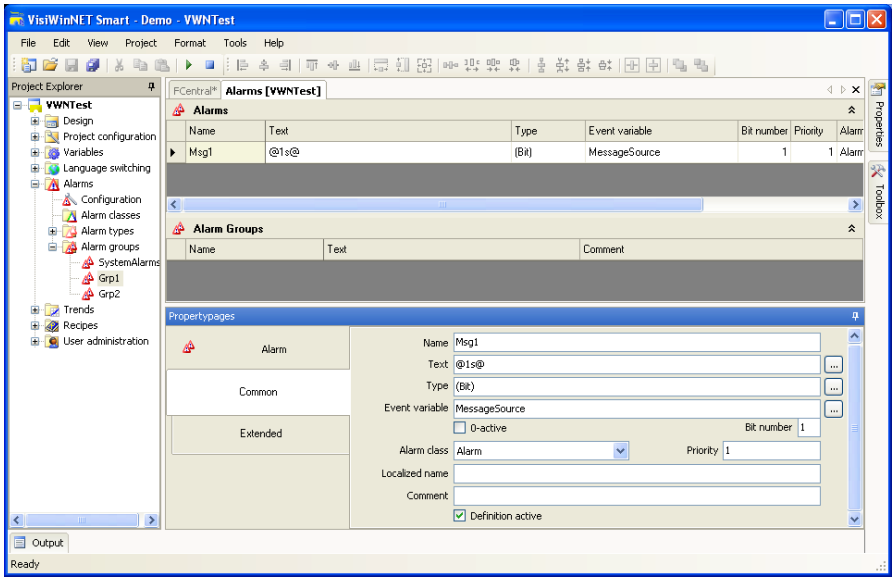
Now, we need two more alarms for each alarm group.

4.7.3 Creating alarms

Alarm for ”Grp1” alarm group

1. In order to create the two alarms in alarm group Grp1, select alarm group **Grp1** in Project Explorer, right-click in the client area and, in the context menu, select the **New** command.
2. Specify the following properties:

Properties for alarms			
No.	**Tab/Property**	**Value**	**Comments**
1	General		
	Name	Msg1	
	Text	MessageSource; Bit 0	Click **[...]**, then click the command button with the **binoculars**, select **Variable MessageSource** and specify **bit number 0**
	Event variable	MessageSource	
	Bit No. of event variable	0	
	Priority	1	
	Extended		
	Status variable	MessageState	
	Bit No.of status variable	0	
2	General		
	Name	Msg2	
	Text	MessageSource; Bit 1	See above
	Event variable	MessageSource	
	Bit No. of event variable	1	
	Priority	2	
	Extended		
	Status variable	MessageState	
	Bit No.of status variable	4	



Alarms for ”Grp2” alarm group

3. In order to create the two alarms in alarm group Grp2, select alarm group **Grp2** in Project Explorer, click in the client area with the right-hand mouse button and, in the context menu, select the **New** command.
4. Specify the following properties:

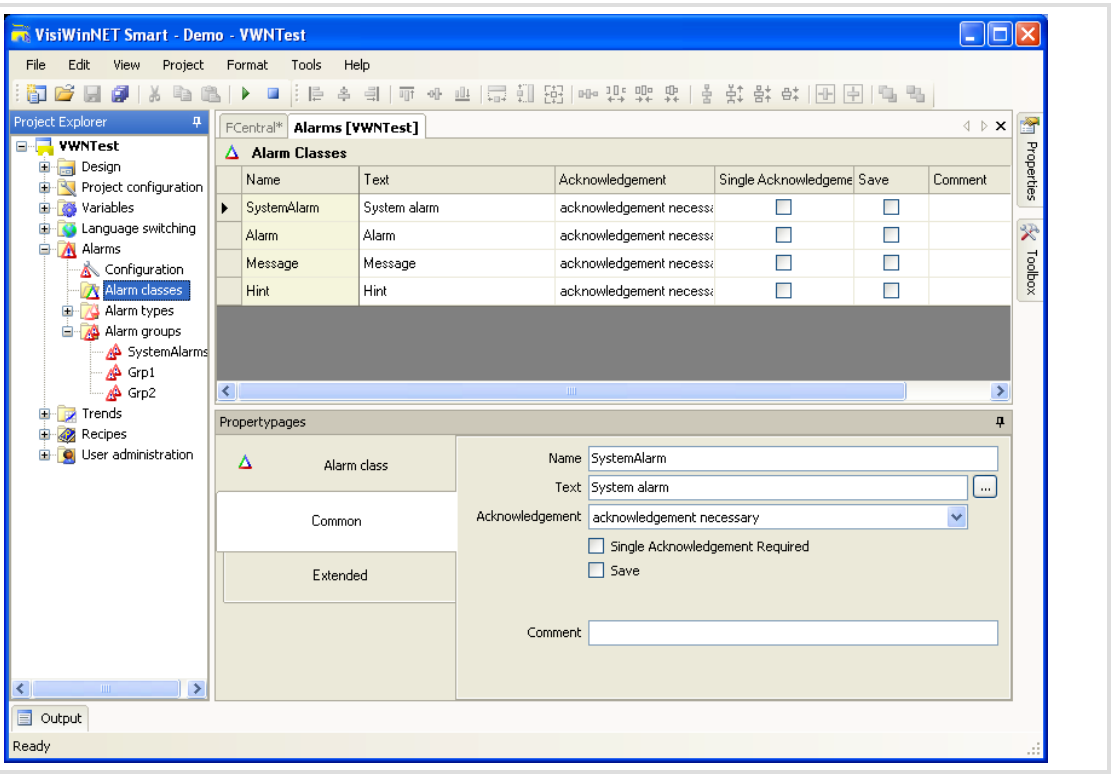
Properties for alarms			
No.	**Tab/Property**	**Value**	**Comments**
3	General		
	Name	Msg3	
	Text	MessageSource; Bit 2	See alarms for alarm group ”Grp1”
	Event variable	MessageSource	
	Bit No. of event variable	2	
	Priority	3	
	Extended		
	Status variable	MessageState	
	Bit No.of status variable	8	
4	General		
	Name	Msg4	
	Text	MessageSource; Bit 3	See alarms for alarm group ”Grp1”
	Event variable	MessageSource	
	Bit No. of event variable	3	
	Priority	4	
	Extended		
	Status variable	MessageState	
	Bit No.of status variable	12	

In the following, the values indicated above will be used to demonstrate all the aspects of how the alarm system functions. The ”Status variable” parameter is optional. Specification of an alarm text is highly recommended. Specification of the event variable is obligatory.

4.7.4 Specifying the type of alarm acknowledgement

The ”Alarm Classes” parameter of all newly created alarms points to ”Alarm”. This is based on a definition hidden under the ”Alarm classes” node of Project Explorer.

1. In Project Explorer, click the node **Alarms** → **Alarm classes**.
   The table of alarm classes appears.
2. Select the **Alarm** line and, on the Properties page, set the **Acknowledgement** parameter to **Acknowledgement necessary**.

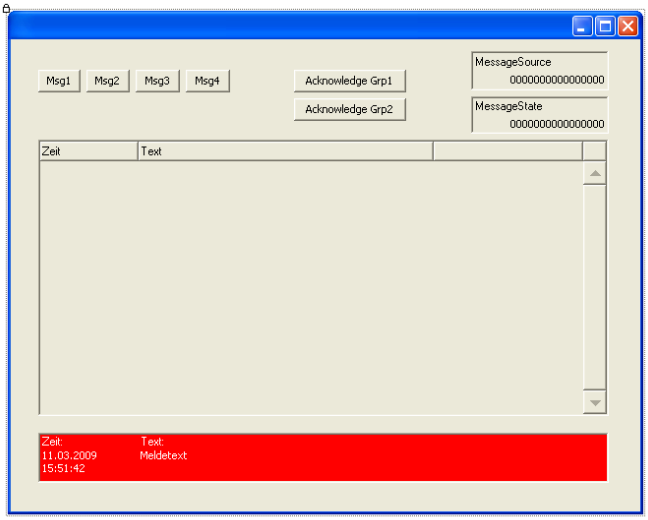


VisiWinNET® Smart supports different types of acknowledgement. One type of acknowledgement defines how often an alarm has to be acknowledged by the user or by the control system before it disappears from the control elements of the application. As a result of this setting, an alarm can assume different states during the runtime. For example, the system distinguishes between the ”occurred” status (event bit has been set) and ”occurred acknowledged” status (after bit setting, occurrence has been acknowledged by the user but the event bit has not yet been reset). The status can be indicated by an icon, a status text and colours. How a status is indicated is also defined in the Alarm classes.

The status of an alarm can also be bit-coded and stored in a variable. For this purposes, the status variables have been defined with bit numbers in the alarms. Three bits represents the status of an alarm in each case.

4.7.5 Visualising alarms

1. Open the **FAlarm** form (📖 180)and place the following control elements there:



Properties for control elements

No.	Control element type/Property	Value	Comments
1	CommandButton		
	Display/Text	Msg1	
	VisiWinNET/VwItem	”MessageSource” variable with bit No. 0	Bit triggers the alarm
2	CommandButton		
	Display/Text	Msg2	
	VisiWinNET/VwItem	”MessageSource” variable with bit No. 1	
3	CommandButton		
	Display/Text	Msg3	
	VisiWinNET/VwItem	”MessageSource” variable with bit No. 2	
4	CommandButton		
	Display/Text	Msg4	
	VisiWinNET/VwItem	”MessageSource” variable with bit No. 3	
5	Key		
	Display/Text	Acknowledge Grp1	
	VisiWinNET/VwItem	”GroupFunction” variable with bit No. 0	
6	Key		
	Display/Text	Acknowledge Grp2	
	VisiWinNET/VwItem	”GroupFunction” variable with bit No. 1	
7	VarOut		
	Display/Label/Text	MessageSource	
	VisiWinNET/VwItem	”MessageSource” variable with ”Binary” format	

Properties for control elements			
No.	**Control element type/Property**	**Value**	**Comments**
8	VarOut		
	Display/Label/Text	MessageState	
	VisiWinNET/VwItem	”MessageState” variable with ”Binary” format	
9	AlarmList		List field; does not need to be adapted
10	AlarmLine		Red line under list field; does not need to be adapted

4.7.6 Testing the alarm system

1. Press **[F5]** to start the project, click the **FAlarm** button and set the four switches **Msg1** to **Msg4**.
   – All four alarms are shown in the alarm list·
   – In the alarm line, the ”Msg4” alarm is shown. This is the currently active alarm with the highest priority in the system.
   – The ”MessageState” variable shows the status ”0100” (binary) for each alarm. If necessary, this status can also be signalled back to the control system.
2. Reset the four switches **Msg1** to **Msg4**.
   – In the alarm list, the alarms are retained. However, the status changes and therefore the colour as well. Due to the type of acknowledgement selected, acknowledgement is necessary in order to finally reset the alarms.
   – The status bits of alarms and alarm groups now show the status ”0010”.
3. Acknowledge the alarms one after the other by double-clicking the corresponding entry in the alarm list.
   – The alarms then disappear from the alarm list.
   – The status bits in ”MessageState” are also reset.
4. Set the four switches **Msg1** to **Msg4** again and then click the two key control elements.
   The contained alarms are all acknowledged together by means of the acknowledgement bit of a group.

4.7.7 Filtering alarm displays

VisiWinNET® Smart makes it possible to filter the alarms displayed in the control elements. As a result, it is possible to configure an alarm list that only shows alarms relating to faults in the plant, for example, and another alarm list that contains notes on production. In the client-server system, it is therefore also possible to show only those alarms that are important for the work station and, nevertheless, to retain the data centrally.

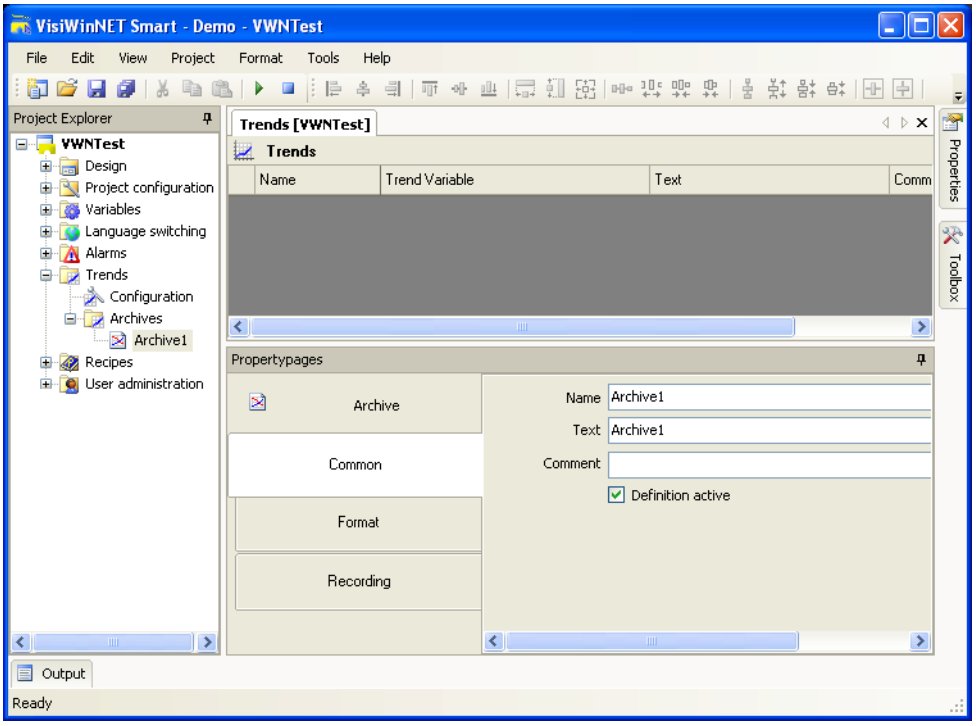
1. Click the **AlarmList** control element on the right at the top on the small triangle.
   The ”AlarmList Task” menu appears.
2. In the menu, click **AlarmFilter (Classes: ; Groups ...)**.
   The ”AlarmFilterItemEditor” dialog opens. The dialog enables the selection of alarm groups and alarm classes. Only the alarms that are in the groups selected here are shown in the control element.
3. Double-click the entry **Grp1**.
   The ”Grp1” alarm group is selected.
4. Click **OK** to close the dialog.
5. Press **[F5]** to start the project and set the four switches **Msg1** to **Msg4**.
   Only the alarms in ”Grp1” are shown in the alarm list. However, the status bits show that the alarms in ”Grp2” have also been generated. The filter is therefore only actually restricted to what is shown in the control element.

4.8 Recording process data (archive system)

The VisiWinNET®Smart archive system allows process values to recorded. The ”TrendChart” control element is available as a component for viewing the values. The necessary definitions as to which process values are to be recorded and how this is to be done are made in the archive editor.

1. In Project Explorer, open the **Trends** node. Then right-click the **Archive** subnode and, in the context menu, select the **New** command.
2. On the Property page of the new archive, define the following parameters:

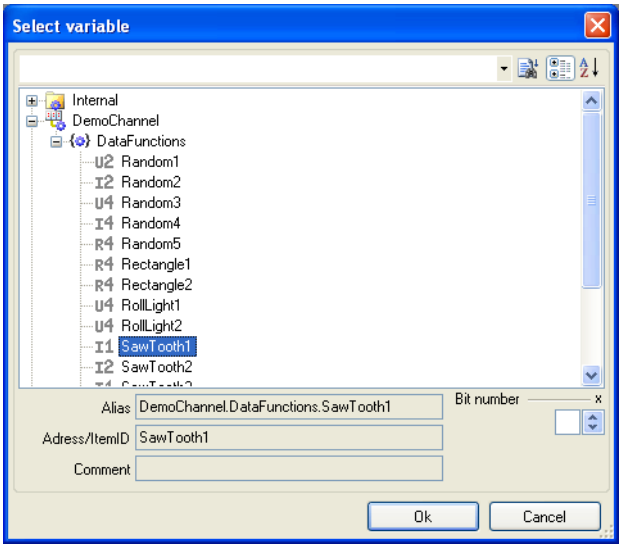
Properties for archive		
No.	**Tab/Property**	**Value**
1	General	
	Name	Archive1
	Format	
	Archiving mode	Subsequent archive (on hard disk)
	Recording	
	Scanning time	10 ms (restrictions for EL 100: 📖 141)



An archive is a container for values that are to be recorded. This container ...·
– places several recorded process values together in files
– determines the scanning behaviour
– manages the generated files
How the process values to be recorded are specified is described in the following step.

3. Right-click in the client area and select the **New** command.
   A new trend definition is created.

4. Click the **[...]** command button in the ”Trend Variable” column to open the ”Select variable” dialog.



5. Select a variable, e.g. ”SawTooth1”.

   The process value has now been specified. Now, it is only necessary to specify how the recorded values are displayed in the application.

6. Click the **Design** node to create a new **FTrend** form. Under **FormBorderStyle**, assign the value **Fixed Dialog** to its properties. (🕮 180)

   The form is then shown as a dialog with the standard Windows command buttons for controlling the size of the window. You can use the buttons to close the dialog and change to the window from which it was called.

7. On the **FCenter** form, create a command button that calls up this dialog (🕮 182).

Properties for control elements

No.	Control element type/Property	Value	Comments
1	CommandButton		
	Display/Text	FTrend	
	VisiWinNET/Command	ShowForm(FTrend)	

8. On the **FTrend** form, place a control element of the type **Trend Chart** (”System” section).

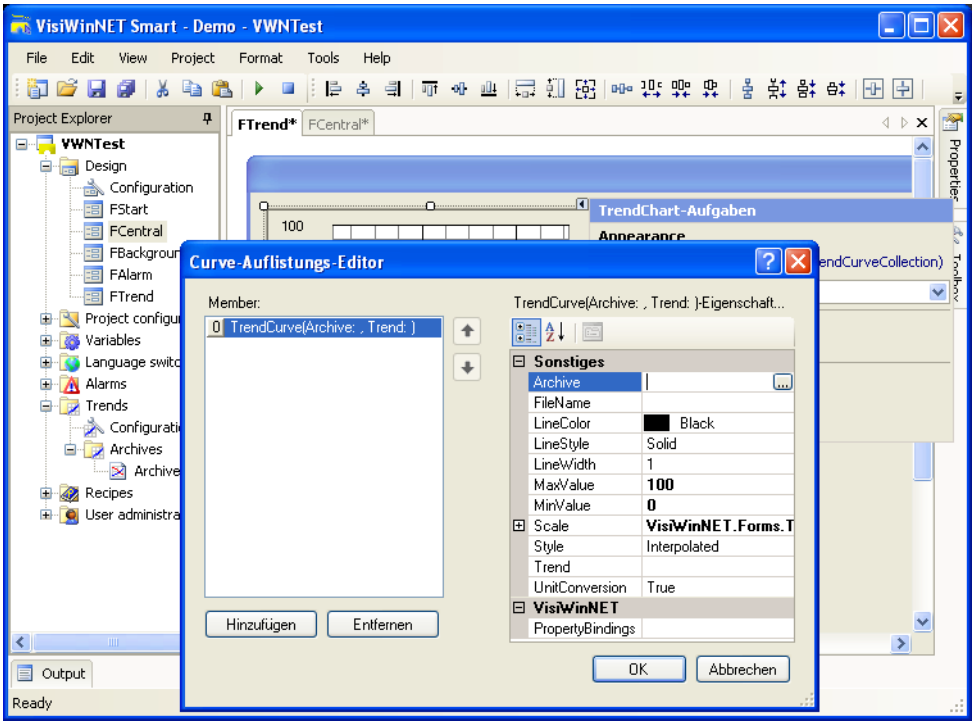
   If you use a mouse-click to position the control element, you then have to make it larger by pulling on it by its ”sizing handle” until the chart is the desired size..

9. In the ”TrendChart tasks” task menu which now appears, click **Curves (VisiWinNet.Forms ...)**.

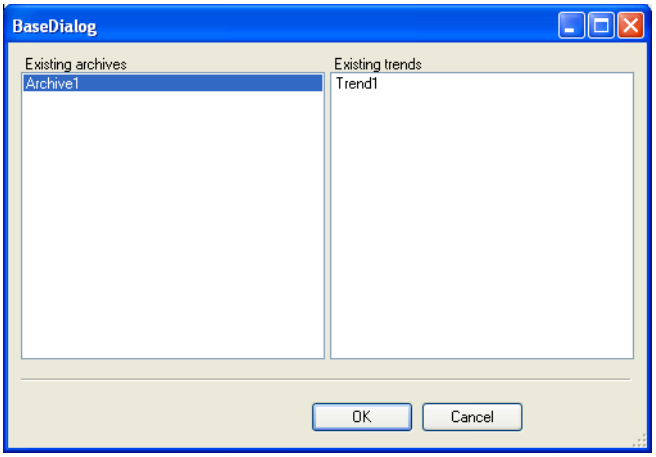
Note!

The tasks menu is closed as soon as you left-click somewhere outside the menu. You can open it again by marking the control element and clicking the small button ▶at the upper edge of the control element.

The dialog ”Curve List Editor” is opened. The curves to be recorded are shown in the list on the left-hand side. A curve has already been inserted in the dialog but it is not yet linked to a trend definition.



10. Select the entry on the left-hand side and, on the right hand side next to the ”Archive” property, click the **[...]**command button.



11. In the following dialog ”BaseDialog”, click the **Archive1**entry and **Trend1**. Then click **OK**.
12. In the property **MaxValue**, enter the value **150**.

    The selection made in the dialog is copied into the ”Archive” and ”Trend” properties. ”MaxValue” parameterises the upper visible value limit for this curve.
13. Close the ”Curve List Editor” dialog by clicking **OK** and close the task menu by means of a mouse-click outside the menu.

14. Select the ”TrendChart” control element and click on the **Properties** tab on the top right-hand side in order to display the Properties window.
15. Expand the **TimeScale** propertyand, in the **Resolution** property below this, specify the value **00:02:00**.

    The ”Resolution” property defines the visible time range in the control element. It is therefore possible to parameterise the x-axis in this way.
16. Press **[F5]** to start the project and then go to the **FTrend** form.

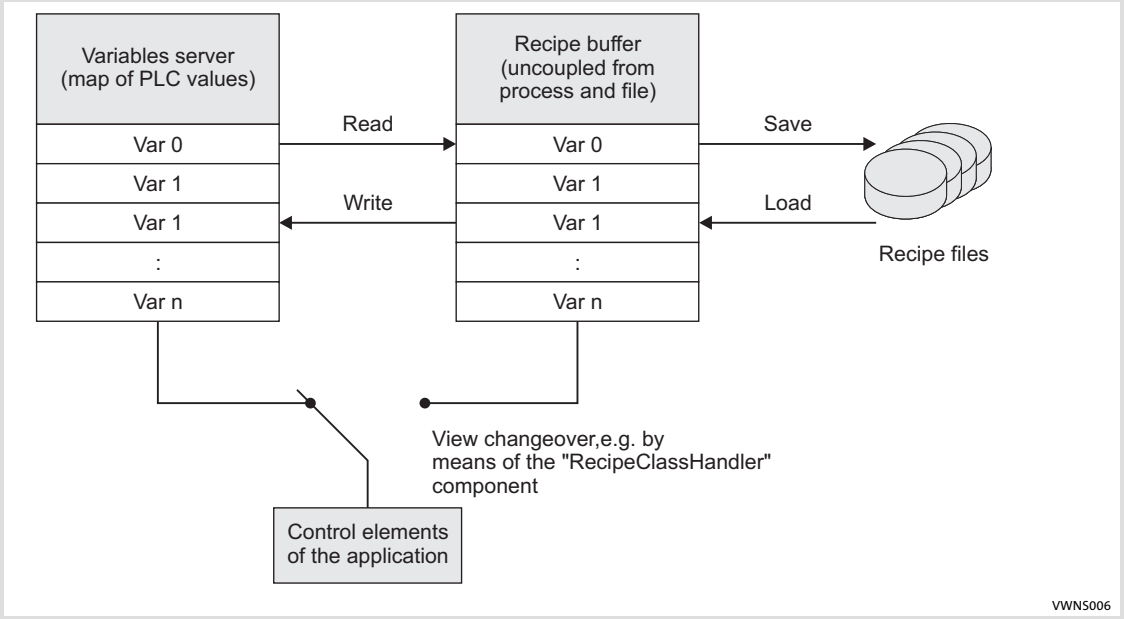
    The application that is running shows the process value indicated in the trend definition, displaying it as a curve.

4.9 Defining visualisation recipes

Visualisation recipes describe sets of process values. Together, they usually determine the settings of a machine for the manufacture of a product from a range of products. It is possible to switch over between the individual products to be manufactured by writing recipes to the PLC. .

During the development time, recipes are defined in order to specify which variables jointly describe the function, i.e. which variables are saved and transferred together.

During the runtime, the sets of process values are stored in so-called recipe files. Between the process image and the recipe files, there is a recipe buffer. This buffer can be used to alter recipe values in the application without affecting PLC data or recipe value data. The recipe buffer is for the parameterisation and optimisation of recipes. Starting from the application, the user loads recipe data from a file, for example, adapts them to the production goal in the recipe buffer and then transfers them to the PLC. The possibility of reversing this process makes it possible to check the set values before a new recipe file is created.



The four transfer functions ”Load”, ”Save”, ”Read” and ”Write” control the individual transfer steps.

Typical input control elements such as ”VarIn” support viewing of the process values in the variables core as well as in the recipe buffer. If these values are to be made available for editing in the application, therefore, new control elements do not have to be added to the application. A view-switching function (for all the variables used in the definition), which is centrally coupled to a recipe definition has to be implemented instead

Other functions such as the recording of recipe value changes in line with FDA or the saving of production comments are easy to parameterise.

Typical tasks for the developer in the recipe system are:

Specifying recipe definitions: Here, which variables describe the recipe are specified. In order to prevent any misunderstanding, please note that this does not concern the values of variables. These are not specified until the runtime.

Adapting designs, editing recipe values: In principle, the existing command elements that are tied to the variables in the process can be used.

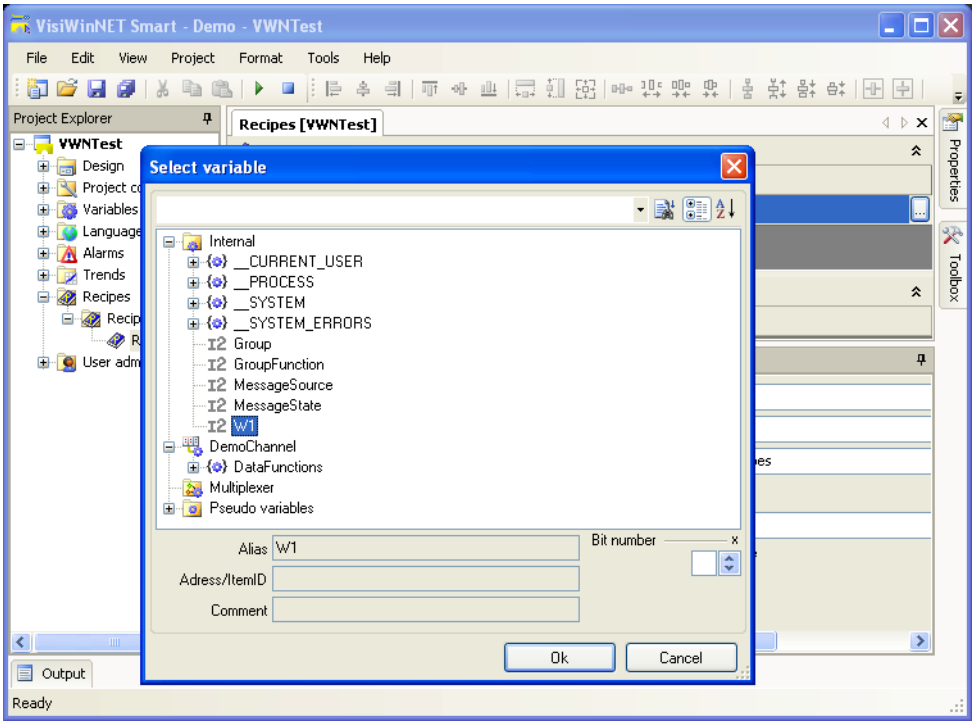
Adapting designs, managing recipe files: Typical tasks such as loading or saving recipe values from files usually require a list of the existing recipe files.

The following example illustrates the above steps with reference to a minimum recipe consisting of two variables.

1. Double-click the **Variables** → **Channels** → **Internal** node in Project Explorer.
   The table editor for defining variables is opened.
2. Press the **[F8]** key to define the following three variables:

Properties for process variables		
No.	**Tab/Property**	**Value**
1	General	
	Alias	w1
	Data type	VT_I2
2	General	
	Alias	w2
	Data type	VT_I2
3	General	
	Alias	Group
	Data type	VT_I2

3. In Project Explorer, open the **Recipes** node.
4. Select the subnode **Recipe classes** and press **[F8]** in order to create a new recipe class.
   The table editor is opened; ”Recipe class1” has been created.
5. In the table editor, right-click in the **Variables** area and select the **New** command in the context menu..
   A new recipe element is the opened.
6. In the line of the new recipe element, click the **[...]** command button.
   The ”Select variable” dialog is opened. Here, the variables that belong to the recipe definition are specified.



7. Select the internal variable **w1** and close the dialog with **OK**.

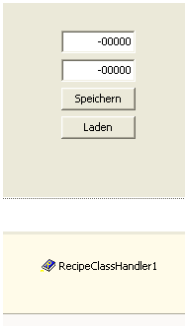
   The highlighted process variable is shown as an element of the recipe in the table editor.

8. Create a new recipe element with the process variable **w2**as described in steps 5 to 7.

9. Go into Project Explorer and open the **Design** node. The, open the **FCentral** form.

10. On this form, place two **VarIn** control elements, two **CommandButton** control elements and a **RecipeClassHandler** control element.

    After the ”RecipeClassHandler” control element has been placed on the form, it automatically slides into the so-called ”component slot” under the Design view of the form.



11. Define the following properties for the control elements:

Properties for control elements			
No.	**Control element type/Property**	**Value**	**Comments**
1	VarIn		
	Display/Text	VarIn1	
	VisiWinNET/VwItem	Variable ”w1”	
2	VarIn		
	Display/Text	VarIn2	
	VisiWinNET/VwItem	Variable ”w2”	
3	CommandButton		
	Display/Text	Save	
	VisiWinNET/Events	The click event is to be logically linked to calling of the ”GetRecipe” function of the ”RecipeClassHandler” component. Click the **[...]** button in the ”Events” property to open the ””Event Processing” dialog. Select the **Click** event and click the **Select** command button. In the function selection dialog which now appears, select the above-named function. After closing the function selection dialog, make sure that the function appears in the function list and that its ”Provider” parameter refers to the ”RecipeClassHandler” component of the form.	
4	CommandButton		
	Display/Text	Load	
	VisiWinNET/Events	The click event is to be logically linked to calling of the ”ShowLoadRecipeDialog” function of the ”RecipeClassHandler” component.	
5	RecipeClassHandler		
	VisiWinNET/Events	● The”GetDoneSucceeded” event is to be logically linked to calling of the ”ShowSaveRecipeDialog” function of the ”RecipeClassHandler” component.· ● The ”LoadDoneSucceeded” event is to be logically linked to calling of the ”SetRecipe” function of the ”RecipeClassHandler” component. After being saved, the results of the events thus parameterised are as follows:· ● The Click event triggers the ”GetRecipe” function. As a result, the values of the variables defined as recipe elements are loaded into the recipe buffer.· ● Conclusion of the reading-in process is signalled by the RecipeClassHandler in the ”GetDoneSucceeded” event. This, in turn, is logically linked to the ”ShowSaveRecipeDialog” function. This is how saving of the recipe file is handled during the runtime, whereby a dialog for entering the name and description appears before saving is carried out. The reverse procedure is triggered by means of the Load command button. First, a selection dialog is used to load a recipe file into the recipe buffer, from where it is then transferred to the process.	
	VisiWinNET/ReceipeClass	RecipeClass 1	The name of the recipe definition from the project database.

This example makes practically no use of the recipe buffer. In many applications, it may be sufficient if recipes are created and optimised directly in the process.

However, use of the recipe buffer make sense if production settings that are not permitted to change the values in the process during the setting procedure are to be made by means of the application. Only after completion of the settings can the data be transferred in one go to the process.

In the ”SmartDemo” project, an example is given of how to use the recipe buffer.

4.10 Printing information

VisiWinNET® Smart contains an interface for printing information from the application. The layout of a printout is defined during the development time by means of special printer forms. The developer places control elements in position and thus determines the information to be printed. During the runtime, the system fills the control elements of the form with the corresponding values. Printout is started by means of the ”PrintHandler” component.

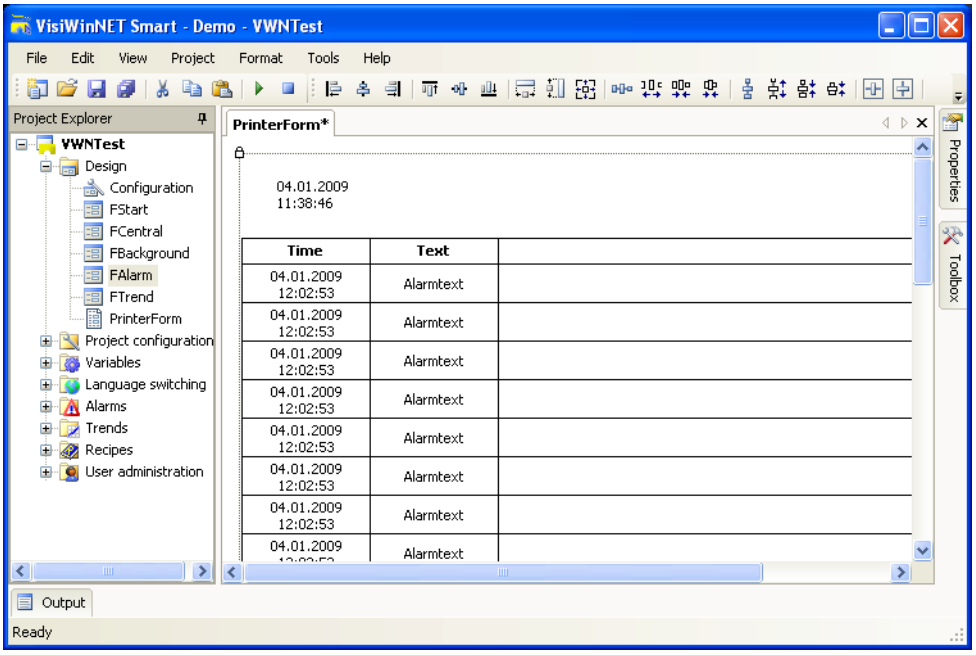
The following example shows how a printing function is implemented in VisiWinNET® Smart.

1. In Project Explorer, right-click the **Design** node and, in the context menu, select the **Add printer form** command.

   A ”PrinterForm” element is created under the ”Design” node and opened in the client area.

   The printer form design after printout is in the form of a sheet of paper. In the ”PageSettings” property, the paper format used (default setting: A4) can be selected.

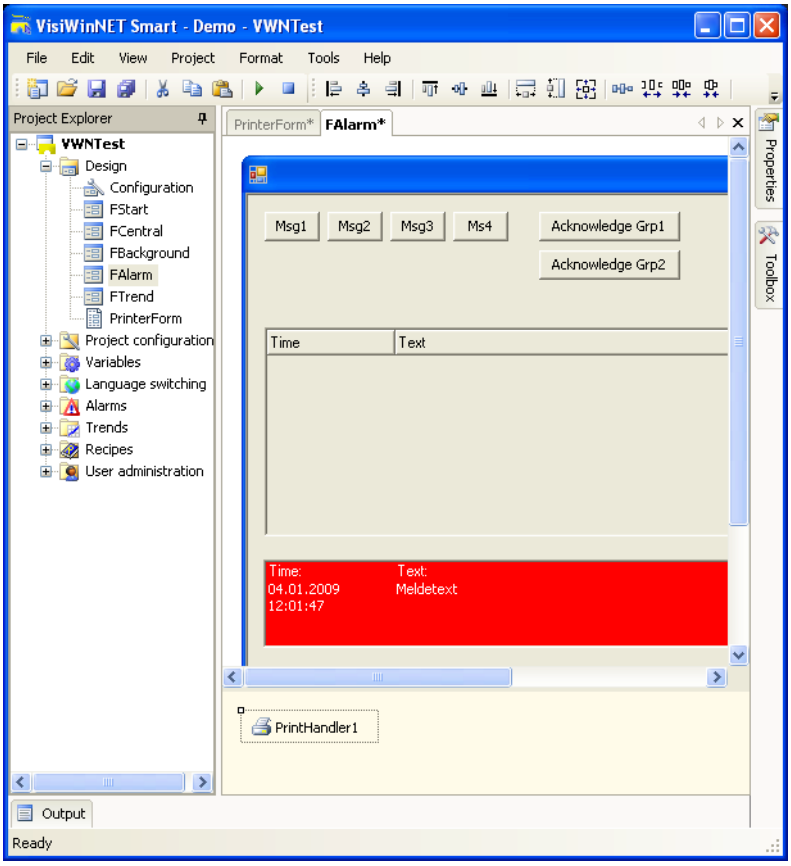
2. From the toolbox section, place a control element of the type **DateTime** and one of the type **AlarmTable** in position on the form.



   The control elements that are available make it possible to specify the layout and the contents of the page to be printed. In order to start printing during the runtime, additional steps are necessary.

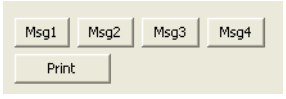
3. Open the **FAlarm** form, go to the **System** section of the Toolbox, and place a control element of the type **PrintHandler** on the form.

   After the PrintHandler control element has been placed on the form, it automatically slides into the so-called ”component slot” under the design view of the form.



4. In the **PrinterForms**property, assign the created printer form to this control element: **Properties window** → **PrinterForms [...]** → **Add** → **"PrinterForm"** → **Add** → **OK**).
5. Place a control element of the type **CommandButton** on the form.
6. On the Property page, specify the following properties:

Properties for control elements			
No.	**Control element type/Property**	**Value**	**Comments**
1	CommandButton		
	Display/Text	Print	
	VisiWinNET/Events	The click event has to be logically linked to calling of the **Print** function of the **PrintHandler1** component.	



By doing so, you logically link the click event to the print function of the PrintHandler component. The PrinterForms property in the component contains a list of printer forms, the contents of which are sent to the printer in the specified sequence.

7. Switch on your standard printer and press **[F5]** to start the project.
   Print forms are always output via the standard printer specified in Windows System Control settings.
8. Change to the **FAlarm** form.
9. Activate one or more of the **Msgx** command buttons.
10. Click the **Print** command button.
    The printer form is then sent to your standard printer.

Note!

- Most Windows CE devices do not support specification of a standard printer. The ”PrintHandler” component therefore contains the ”ShowPrinterSettings” function. In cases of doubt, this function must be called once before the first print job in order to enter the printer settings.These settings then are stored in the project directory. All subsequent print jobs are sent to the printer that has thus been specified.
- Under Windows CE, all PCL-compatible printers can be used.

5 Configuration

5.1 Projecte settings

VisiWinNET® Smart projects contain settings that have an effect on the runtime behaviour of the project. In some cases, these settings are offered as soon as a project is created. Other settings can be altered later during project configuration.

The settings of the project configuration can be reached via the node of the same name in Project Explorer. Project Explorer divides the properties up into different areas. Clicking an area causes the corresponding settings to be displayed in the VisiWinNET Property page.

The available settings depend on the type of the project.

5.1.1 Global project settings

Access: Project configuration → Global

Index card on Properties page: Common

The Common project properties reflect the settings made in the Project Manager when the project was created.

It is not possible to alter the settings shown here. The following standard task is performed by other parts of the Project Manager.

► Renaming of the project: This function is available in the Project Manager.

5.1.2 Common project properties

Access: Project configuration → Global

Tab on Property page: Extended

The extended properties enable you to save general information in the project database.

5.1.3 Selection of the systems

Access: Project configuration → Server components

Tab on Property page: Common

The systems selected here are shown as editors in the integrated development acknowledgement. The editors are used to write the system-specific definitions into the project database.

During the runtime, only the system servers selected here are loaded, i.e. if the trend system is deselected, the recording functions of the trend server, for example, are not active.

5.1.4 Extended project properties

Access: Project configuration → Client (standard type of project) or Project configuration → Runtime (Compact type of project)

Tab card on Property page: Compatibility

This project property contains settings that describe changes in behaviour compared to Versions 5.x and 6.0.

Setting	Description
”Limit” event in ”VarIn” control element extended	This setting relates to the occurrence of the ”Limit” event in the ”VarIn” control element. The event occurs in the control element during input if a limit monitoring function has been activated by means of the ”LimitCheck” or ”UseItemLimits” properties. A choice between two kinds of behaviour is to be made, especially if multiple-digit input values have to be checked: ● **deactivated** (compatible with Versions 6.0, 5.x): The ”Limit” event occurs immediately during input if the specified value exceeds the upper limit. If the value is below the lower limit, the ”Limit” event is prevented from occurring until the input has been marked as completed due to a change of focus or due to the Enter key being pressed. If the lower limit has been set to ”20”, for example, the user can enter the value ”23” by typing in the digits ”2” and ”3” without triggering the ”Limit” event. ● **activated** (standard behaviour with Version 6.1 and higher): If the lower limit has been set to ”20”, for example, the ”Limit” event is triggered when the user enters the value ”23” by typing in the digits ”2” and ”3”. The event occurs after the digit ”2” has been entered because this value is below the lower limit. This setting is applied in practice, e.g. by the entry of digits with the on-screen keyboard (NumPad). In this context, an incomplete entry or an entry that exceeds/falls below the limits is signalled by a colour.
”DecPoint” property writable in the VisiWinNET item	This setting relates to all VisiWinNET control elements that link-up an analogue value as a ”process variable” in the VWItem property. For the link-up of analogue values, the ”DecPoint” property is available within the ”VWItem” property. The function of this property is controlled by the ”DecPoint Property” setting in the VisiWinNET item: ● **deactivated** (compatible with Versions 6.0, 5.x): If the ”UnitConversion” property has been set to TRUE in the control element, ”DecPoint” is inactive during the runtime. ● **activated** (standard behaviour with Version 6.1 and higher): The displacement of the decimal point as specified by means of ”DecPoint” is also applied, if necessary, to the unit conversion function activated by means of ’UnitConversion’. This means that decimal places in the case of numbers with floating decimal points can be specified as well as conversion by means of a unit class.
Optimised activation of variables when a changeover is made from one form to another	The variables needed on forms are reported together to the variables core once the form has been completely loaded. This increases the page-changing speed because the necessary variables for the associated communication channel are activated (deactivated if no longer necessary) together and therefore faster. However, a consequence of activating this option is that the value from the communication channel is not always current in the first ”Change” result of an item (cause: ChangedByConfig) but may be an older value or the starting value. Deactivation of this setting restricts the function of the following components: ● VisiWinNET.DataAccess.MultiItemActivator (implemented in VisiWinNET.Standard.Client/ VisiWinNET.Compact.Systems) ● VisiWinNET.Forms.BaseForm (implemented in VisiWinNET.Standard.Forms/ VisiWinNET.Compact.Forms) ● In VisiWinNET SMART, the forms directly inherit from ”VisiWinNET.Forms.BaseForm”. Here as well, deactivation of the setting results in a functional restriction.

5.1.5 Runtime behaviour

This Configuration dialogue only exists in the ”Standard” types of project.

Access: Project configuration → Runtime

Tab on Property page: Standard

Contains settings of the runtime and of the VWNManager

Setting	Description
Time settings	Cycle time settings for the VWEngine. The cycle time itself (if all the data for a cycle have arrived faster than desired, the system cycle pauses until the set cycle time has expired; if data acquisition lasts longer than the time set, the system cycle automatically becomes longer, i.e. no request for data is lost) as well as the idle time between two cycles (is used for reporting the computing time to the visualisation application or other processes) can be set. **ms cycle time:** The system cycle time for data acquisition; a basis cycle of 1 sec. (100 ms) has been set as the default setting.. **ms Idle time:** The idle time between two system cycles (default setting 1 ms)
VisiWinNET Manager	The main menu of the VisiWinNET Manager can be configured. **Configuration deactivated:** The Configuration dialogue cannot be selected during the runtime. **Variables dialogue deactivated:** The VWEngine dialogue cannot be selected during the run time. **Start/Stop/Exit deactivated:** The Start, Stop and Exit menu items cannot be selected during the runtime.
Runtime data	**Encode data:** This specifies whether data generated from the ”Protocol”, ”Recipe” and ”Alarm” systems during the runtime are to be encoded. This setting ensures greater protection against manipulation of the generated data. **Password for write access:** This specifies the password for Access databases generated from the ”Protocol”, ”Recipe” and ”Alarm” systems during the runtime. If the password specified here is not entered when the database is opened (e.g. with Access), the file can only be opened in write-protected mode.
Communication	**Waiting time for VisiWin driver:** This specifies the length of time (in seconds) for which the variables core waits for a VisiWin driver in the initialisation phase. If the driver does not start in the time specified here, an error message is generated. **Read all variables 1x on starting:** This specifies whether the variables core reads all process values once during the initialisation phase. The purpose of this one-off reading process is to initialise the internal variables buffer.

5.1.6 Language settings

Access: Project configuration →Global

Tab on Property page: Languages

The dialogue lists the languages contained in the project. One of the languages shown here can be selected as the developer language. The functions of the developer language are as follows:

- Language-switchable texts used in the forms are displayed in the developer language during the development time.
- Language-switchable texts such as an alarm text for example are assigned in the respective editors when the developer language is entered.

5.2 Communication

The VisiWinNET®Smart scope of supply includes several example projects that have been copied to your hard disk during the program installation.

These projects can be used to get to know VisiWinNET®Smart and, moreover, to check the PC↔operating unit EL1xx connection and the readiness for operation of the operating unit.

Prepare the following to control e.g. a servo inverter by means of the EL1xx and the example project:

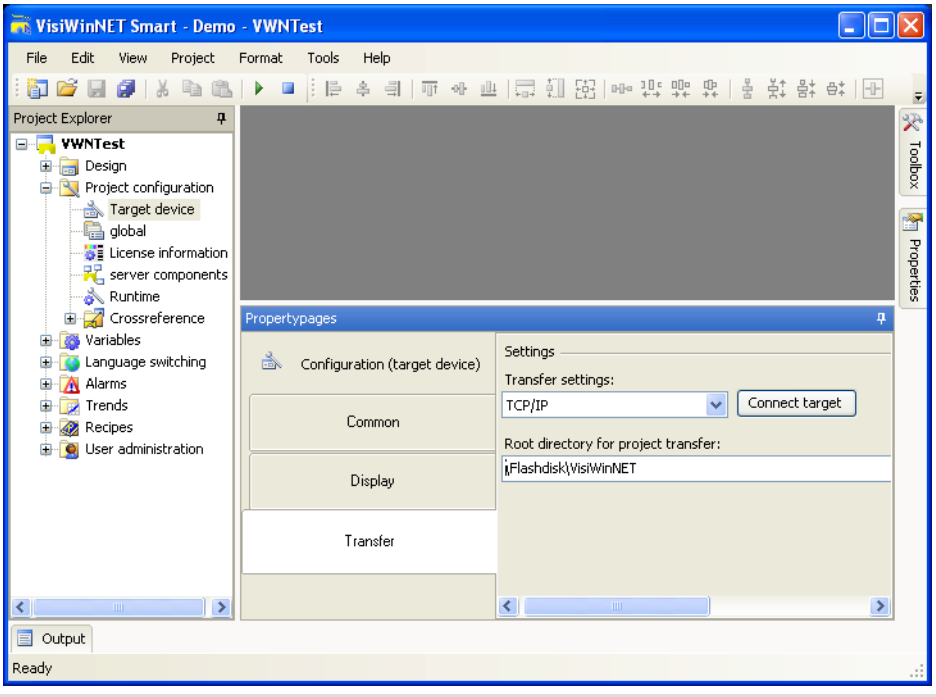
Drive	What to do	Procedure
9400 servo inverter	• Make the basic configuration and select the "Actuating drive - speed" application • Assign CAN node address 2 (C0350) • Transmit project to the controller • Enable controller and activate fixed setpoint 1	🕮 Engineer Online Manual

5.2.1 Transfer project via LAN to EL1xx

Precondition: The network connection between the PC and EL1xx has been set up and is active (🕮 User instructions for the HMI EL1xx, chapter entitled "Setting up an Ethernet connection"). An application running on the target device must be ended.

Proceed as follows:

1. Go into Project Explorer and open the folder **Project configuration**and then open the subfolder **Target device**.
   The "Configuration (target device)" Properties page is shown.
2. Open the **Transfer** tab.



3. Selection the TC/IP connection in the selection field **Transfer settings**.

4. Select the TC/IP connection in the selection field **Transfer settings**.
   In order to change the connection parameters, click **Connect target**. The IP address of the EL1xx is shown if you double-click the network system in the status line at the EL1xx.
5. In the field ”Root directory for project transfer”, specify the download destination for the runtime program on the EL1xx. For example: \Flashdisk\VisiWinNet. Possible download destinations:
   \Flashdisk\xxxx: Flashdisk (program is retentive)
   \Temp: RAM (program is volatile)
   \Storage\xxxx: SD card (program is retentive and portable; is only offered if an SD card has been plugged into the EL 1xx)
   \Hard Disk\xxxx: USB memory medium (like SD card)
6. Click the ▶(”Start project”) button in the toolbar and following the instructions in the dialogs.
   A dialog is shown in which you can edit the previously allocated IP.
7. Click **Accept**.
   The project is transferred to the target device.

5.2.2 Configure CAN connection via codes

An EL1xx CAN can communicate with a CAN fieldbus system via the CAN interface.

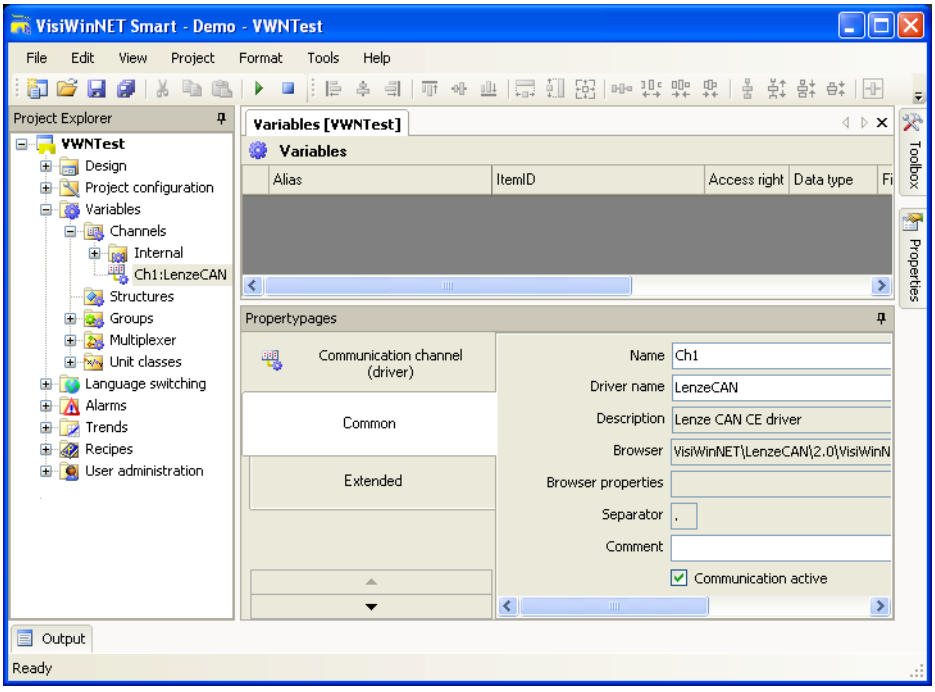
Configure channel

Prerequisite:

- The EL1xx is connected via LAN to the PC on which VisiWinNET® Smart has been installed.
- The EL1xx is connected to the controller via CAN.
  For more information, please read the documentation for the EL1xx and the controller.

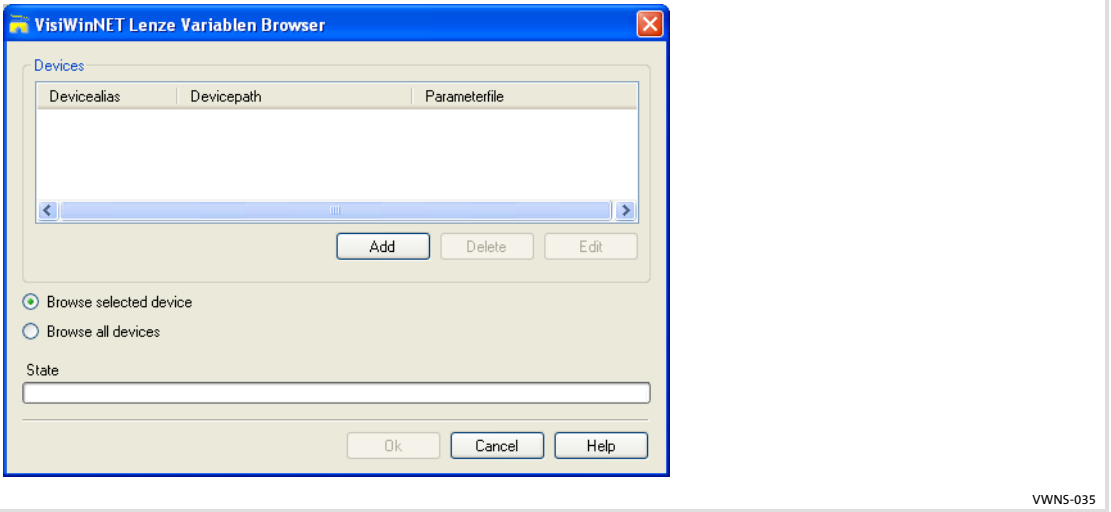
Proceed as follows:

1. Go into Project Explorer and create a new channel: Highlight the **Variables / Channels** folder, right-click it and then select **New**.
2. In the line **OPC-Server/Driver**, click **[...]** and then select the **LenzeCAN** communication driver.
3. Close the dialog with **OK**.
   The channel is created and displayed in the project tree.
4. Double-click the new channel in the object tree and configure it on the Properties page.

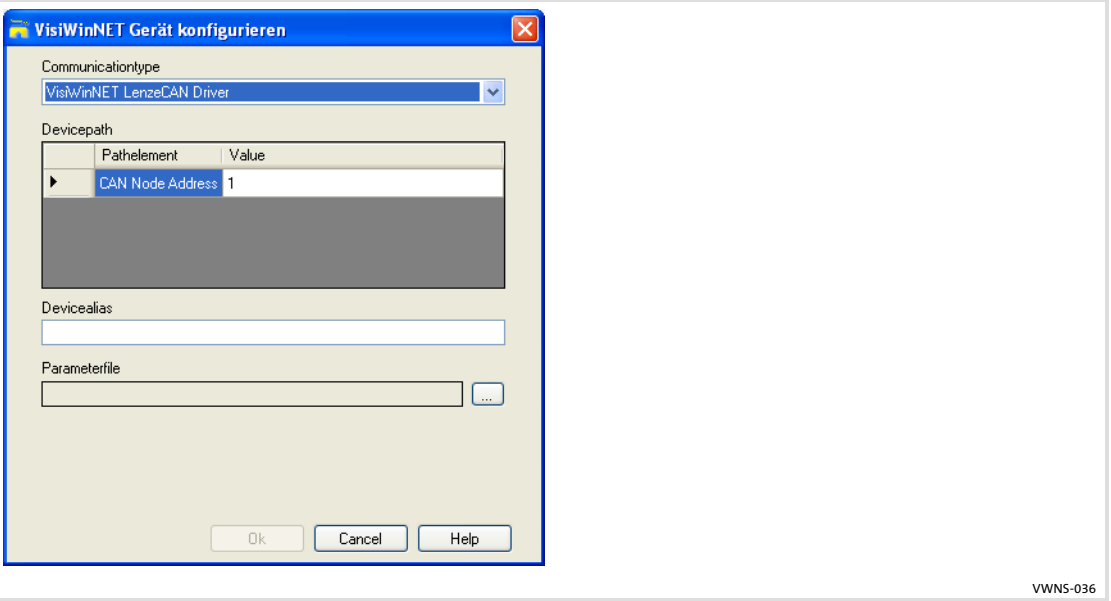


Browsing variables

5. Highlight the **CAN channel** in Project Explorer, right-click it and select **Browse variables**.



6. Click **Add** and fill in the "Configure device" dialog.



Communication type: Communication type "VisiWinNET LenzeCAN Driver"

Device path: CAN node address of the fieldbus node with which data is to be exchanged.

Parameter file: EDS/GDC file for the import of variables

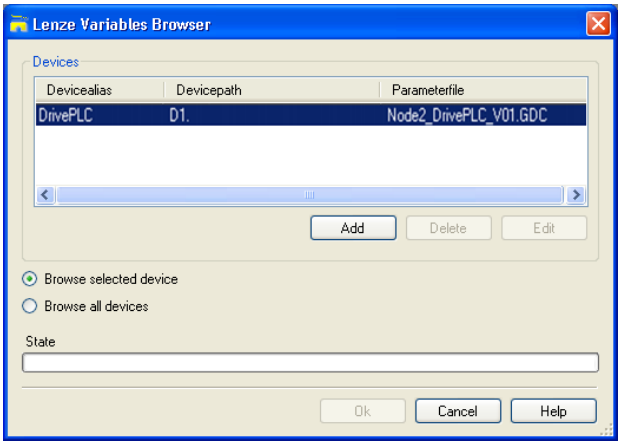
For importing codes via a GDC file, you always need a pdb file that matches the device for which the GDC file has been created. The pdb files are stored in the main GDC directory in the folders pdb/049 (German) and pdb/044 (English).

Device alias: Enter any name you like.

7. Close the dialog with **OK**.

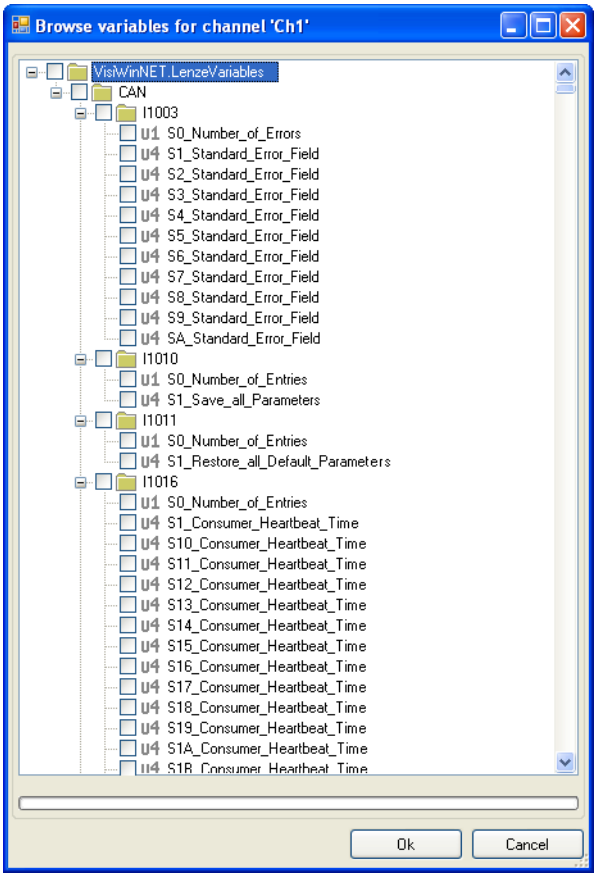
Reading in variables

8. Select the device from which variables are to be read.



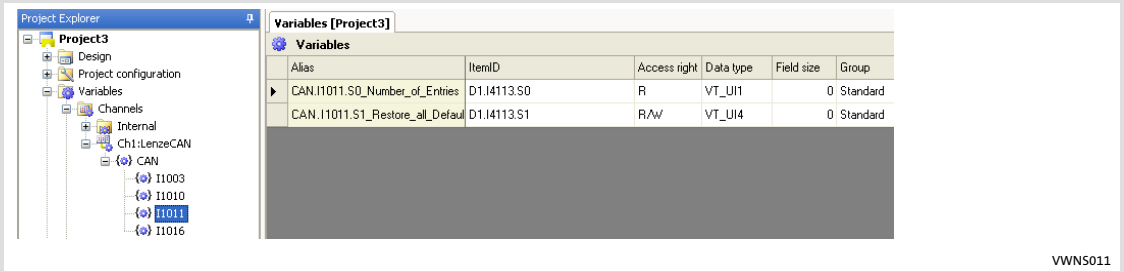
VWNS005

9. Close the dialog with **OK**.
10. In the following dialog, highlight the variables to be imported.



VWNS010

11. Close the dialog with **OK**.
    The selected variables are now available in VisiWinNET® Smart.



5.2.3 Configure CAN connection via PDOs

In addition to accessing codes, the driver can also manage CAN process data (PDOs). A total of 10 PDOs can be set with Rx or Tx function. You can choose between ”sync”, ”event-controlled” and ”cyclic” transmission. (For more information, please see the Lenze Communication Manual.)

The following example describes how to configure PDOs.

Task: A byte value is to be cyclically (time interval 1 s) output by the EL1xx with the identifier ”700dec” on the CAN bus. A Tx-PDO is always output with a length of eight bytes. In this example, the byte is to be output in telegram byte 1.

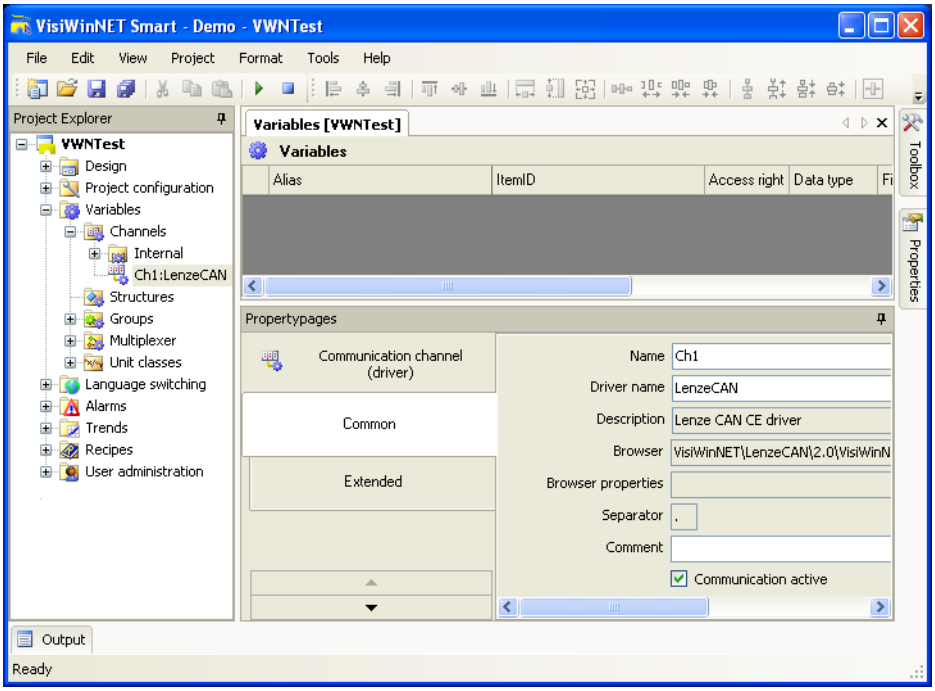
Configure channel

Prerequisite:

- ► The EL1xx is connected via LAN to the PC on which VisiWinNET® Smart has been installed.
- ► The EL1xx is connected to the controller via CAN.
  For more information, please read the documentation for the EL1xx and the controller.

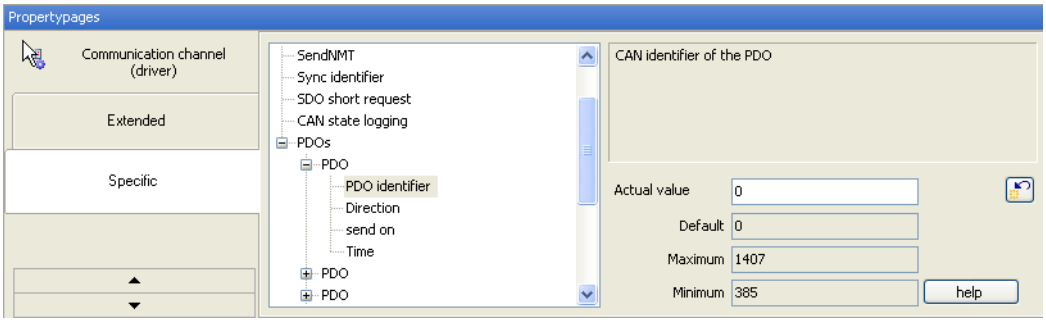
Proceed as follows:

1. Go into Project Explorer and create a new channel: Highlight the **Variables / Channels** folder, right-click it and then select **New**.
2. In the line **OPC-Server/Driver**, click **[...]** and then select the **LenzeCAN** communication driver.
3. Close the dialog with **OK**.
   The channel is created and displayed in the project tree.
4. Double-click the new channel in the object tree and configure it on the Properties page.



PDO-specific settings

5. Activate the **Specific** tab and open the node **PDOs/PDO/PDO Identifier**.



Enter the following values:

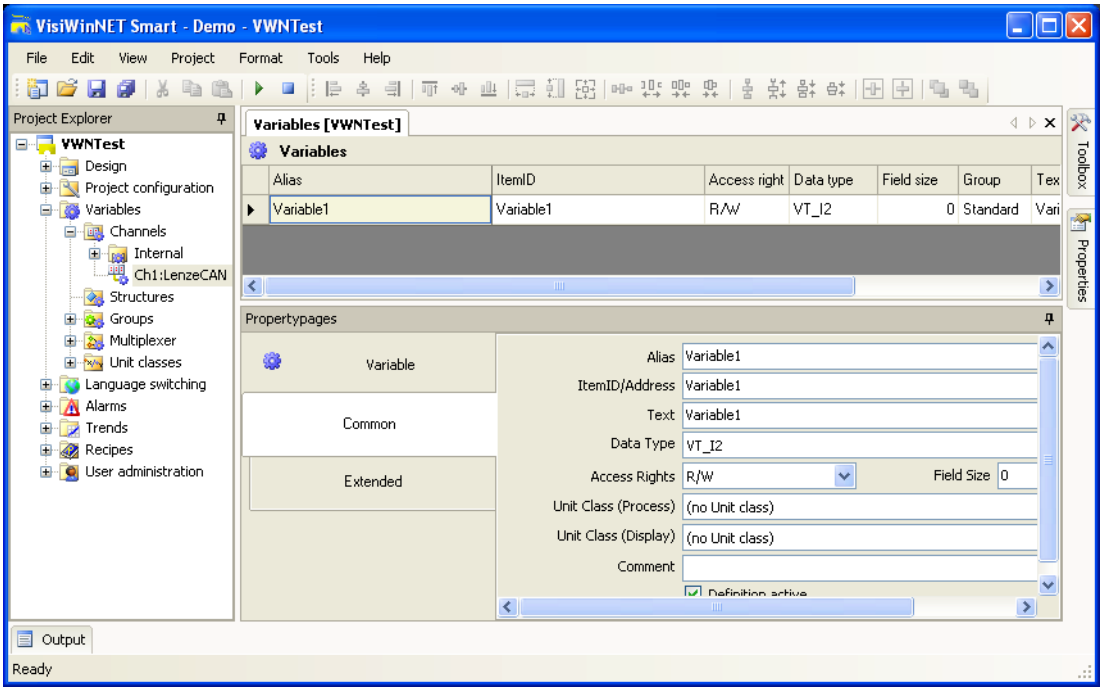
PDO identifier: 700_{dec}
Direction: Tx
Send on: Timer
Time: 1000ms

All the necessary settings regarding PDO behaviour have now been defined. After the PDO-specific settings have been made, it is necessary to create the PDO as a VisiWinNET variable for display purposes or for the entry of values.

Creating PDO variables

6. Click the client area with the right mouse button and select **New** from the context menu.

   A variable is created.



7. Configure the variable as follows:

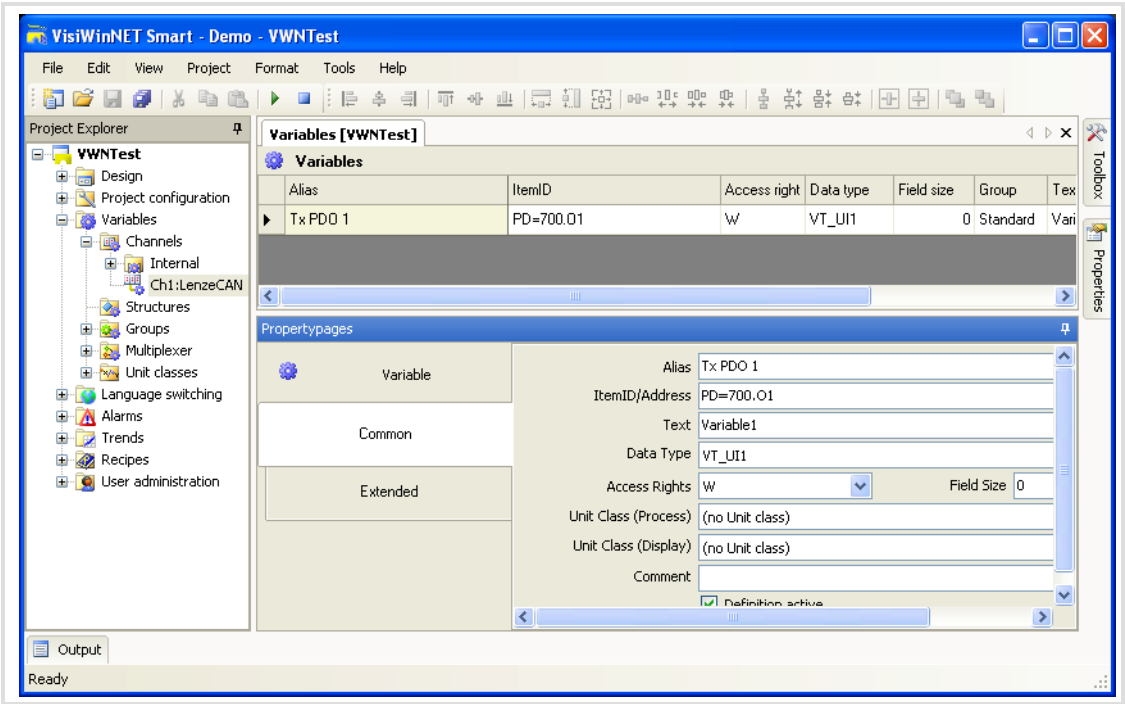
   Alias: Under the alias, the variable is made available for further processing in VisiWinNET. In this example, the name ”Tx PDO 1” is chosen as the alias.

   ItemID/Address: The item ID is used to establish the connection to the PDO. The following syntax must be used for the entry: PDOxxx.Oy.

   The PDO identifier is entered for xxx. The y stands for an offset in the PDO telegram, i.e. in the event of a 0 offset, the variable will be mapped from byte ”0” onwards. In respect of this example, where the identifier has been defined with 700 and the byte variable in byte 1 is to be mapped, the entry for the ItemID/Address must be as follows: PDO700.O1

   Data type: Definition of the data type; here ”VT_UI1”

   Access Rights: Specifies the type of access rights; here, ”W” (Write)



The variable Tx PDO 1 can now continue being used as an item (e.g. as the ”VarIn” input field).

5.2.4 Configure MPI connection

Via the MPI interface (multi-point interface), an ”EL1xx MPI” can communicate with an S7 PLC.

Configuring a channel

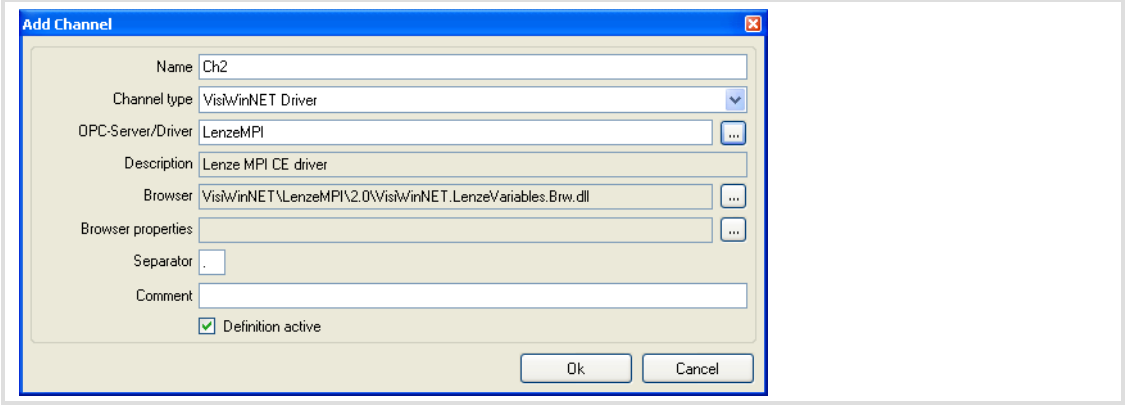
Prerequisite:

- The EL1xx is connected via LAN to the PC on which VisiWinNET® Smart has been installed.
- The EL1xx is connected to the PLC via MPI.
  For more information, please read the documentation for the EL1xx and the PLC.

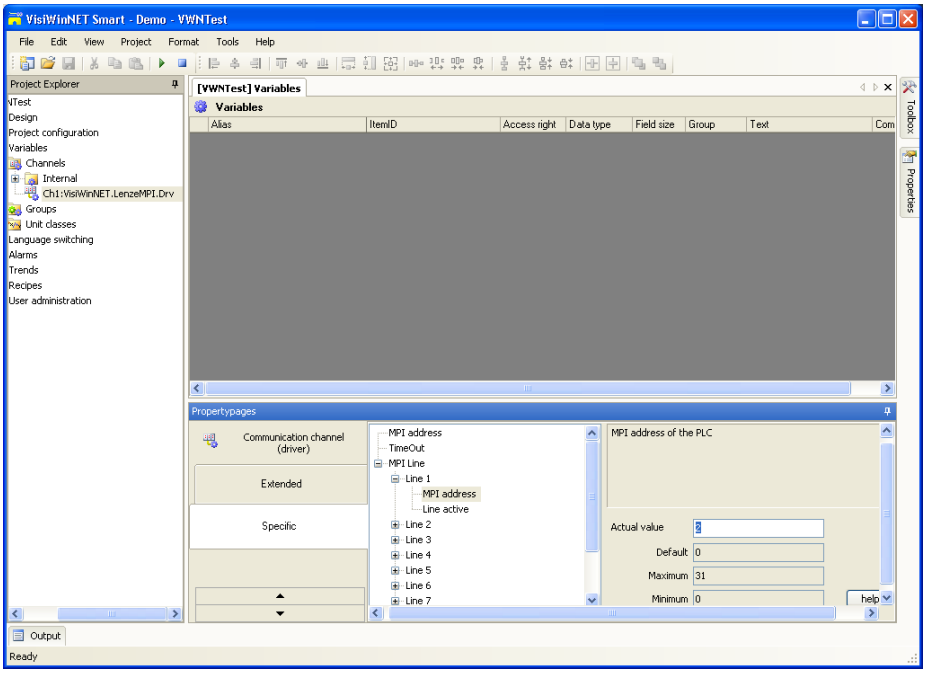
Proceed as follows:

Prerequisite:

1. Create a new channel in Project Explorer: highlight the **Channels** folder, right-click it and select **New**.
2. In the line **OPC-Server/Driver**, click **[...]** and select the communication driver **LenzeMPI**.



3. Close the dialog with **OK**.
   The channel is created and displayed in the project tree.
4. Double-click the new channel in the object tree and configure it on the Properties page.



MPI address: This is where to set the address of the EL1xx. The address must be identical with the device address selected in the MPI-Applet of the EL1xx.

Time-out: Possible setting/time-out behaviour. Default setting: 1 s.

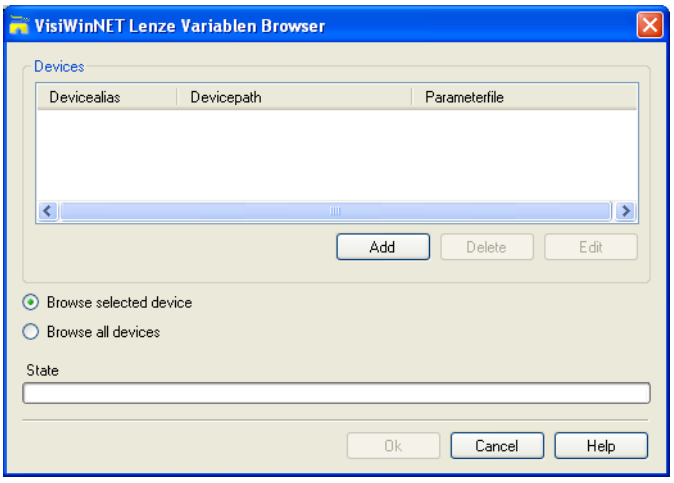
MPI Line: A communication relationship between the EL1xx and a PLC is called an MPI line. A maximum of 10 lines can be set up; i.e. an EL1xx can exchange date with a maximum of 10 PLCs per MPI.

MPI address: MPI address of the PLC.

Line active: Activates or deactivates an MPI line. Selecting ”1” in the ”Set value” field activates the line.

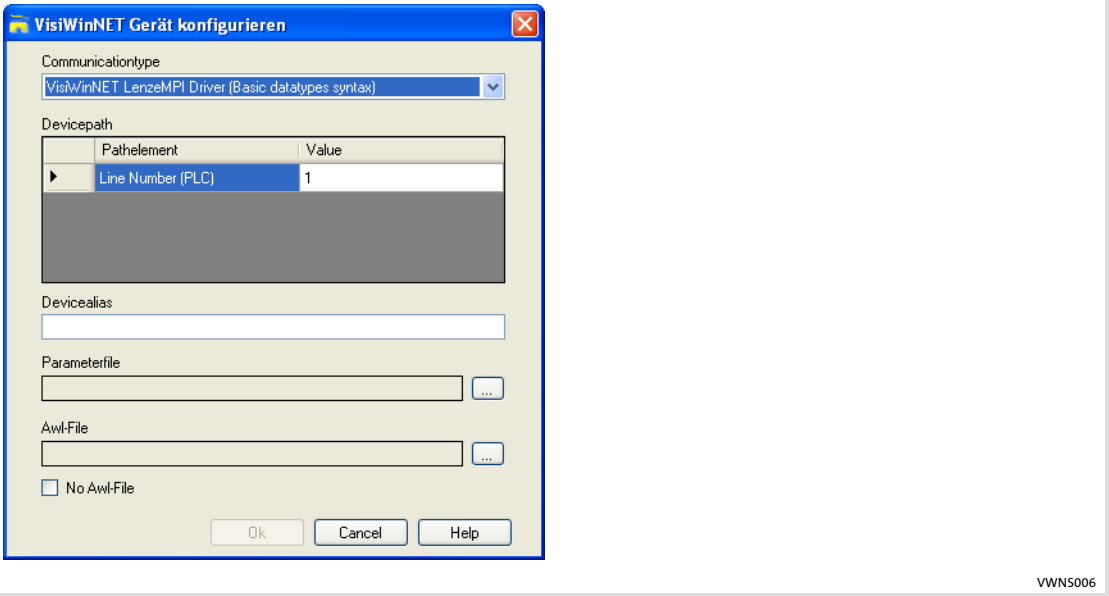
Browse variables

5. in Project Explorer, click **MPI channel** with the right mouse button and select **Browse variables**.



VWNS005

6. Click **Add** and fill in the ”Configure device” dialogue.



Communication type: Communication type ”VisiWinNET Driver (basic datatypes systax)“ or ”VisiWinNET LenzeMPI Driver (instruction list syntax)“.

Device alias: Any name

Parameter file: sdf file for the import of variables from an S7 symbol file (📖 246).

Instruction list file: Instruction list file to be read (📖 247).

No instruction list file: Activate this checkbox, if no instruction list file is to be read. Otherwise, the dialogue cannot be completed without assigning an instruction list file.

For browsing the variables, you need sdf files (generated from the symbol table) or instruction list files (generated from data blocks) which can be generated by Step7. In the following, files which have already been generated will be imported.

Note!

The selection of ”Basic data types syntax“ or ”Instruction list syntax” only affects the representation of the item ID of the data block variables.

The following example is used to describe the difference for an M0.0 flag:

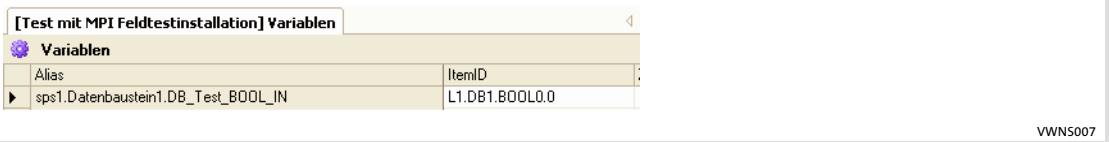


Fig. 5-1 Representation according to „Basic data types syntax“

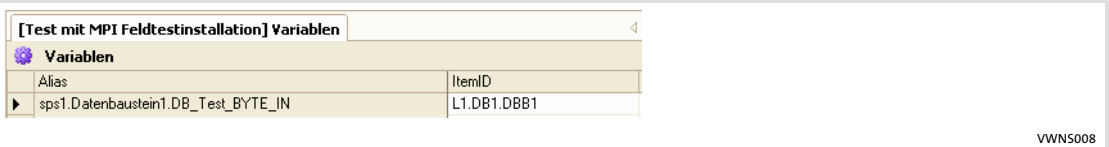
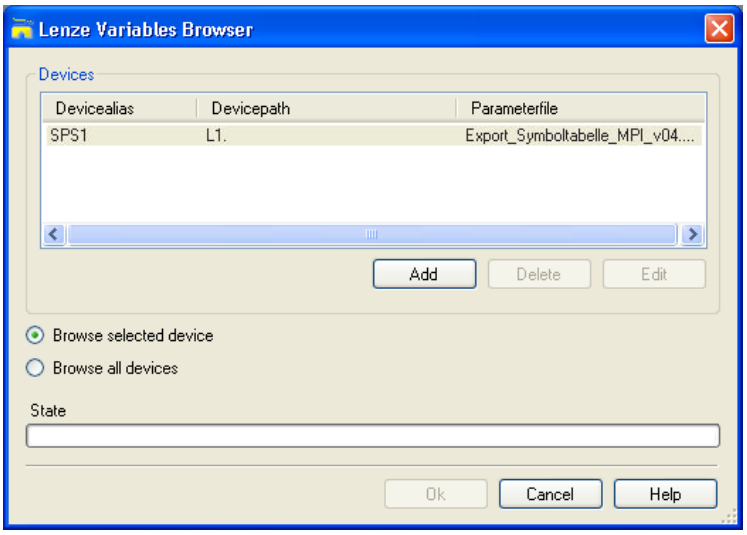


Fig. 5-2 Representation according to „Instruction list syntax“

7. Close the dialogue with **OK**.

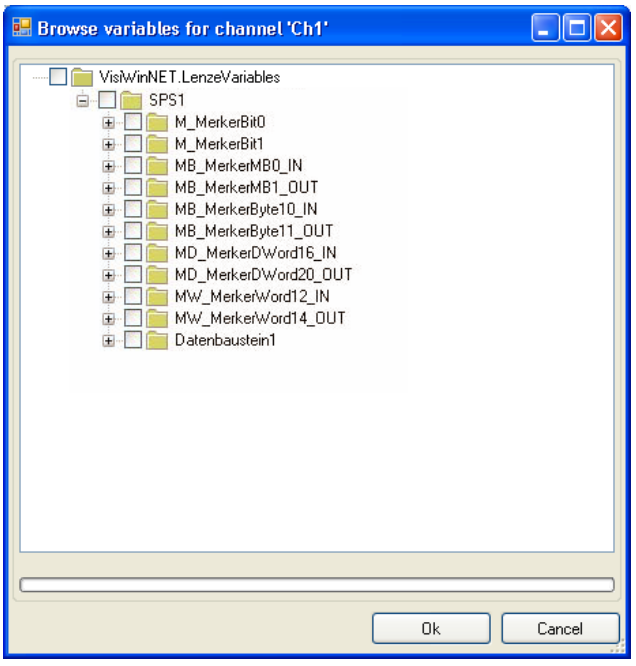
Read variables

8. Select the device from which variables are to be read.



VWNS009

9. Close the dialog with **OK**.
10. Select the variables to be imported from the following dialog.



VWNS-010

11. Close the dialog with **OK**.
    The selected variables are now available in VisiWinNET® Smart.

5.2.5 Configuring the L-force Logic (CoDeSys) connection

Via VisiWinNET®Smart V 6.2 (or higher), an ”EL1xx PLC” can communicate with an L-force Logic (CoDeSys) application.

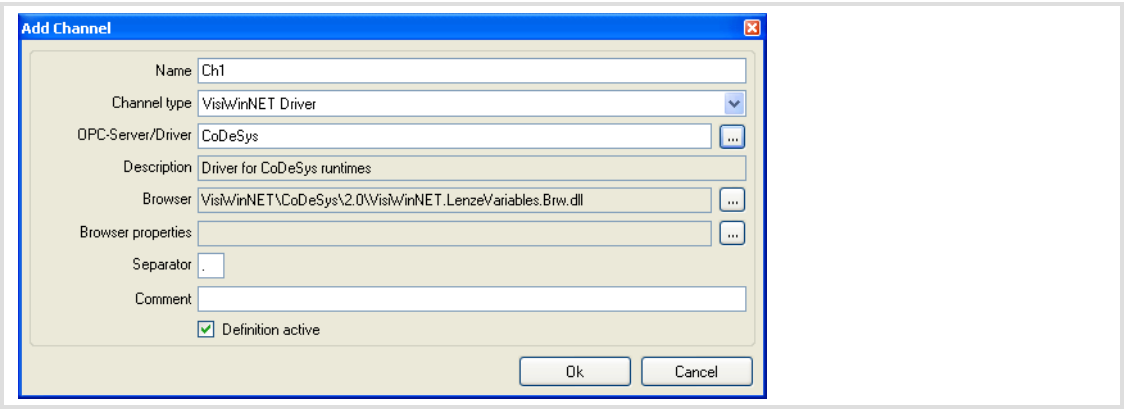
Precondition:

► VisiWinNET®Smart and L-force Logic (CoDeSys) have been installed on the same PC or on different PCs that are connected to each other via LAN.

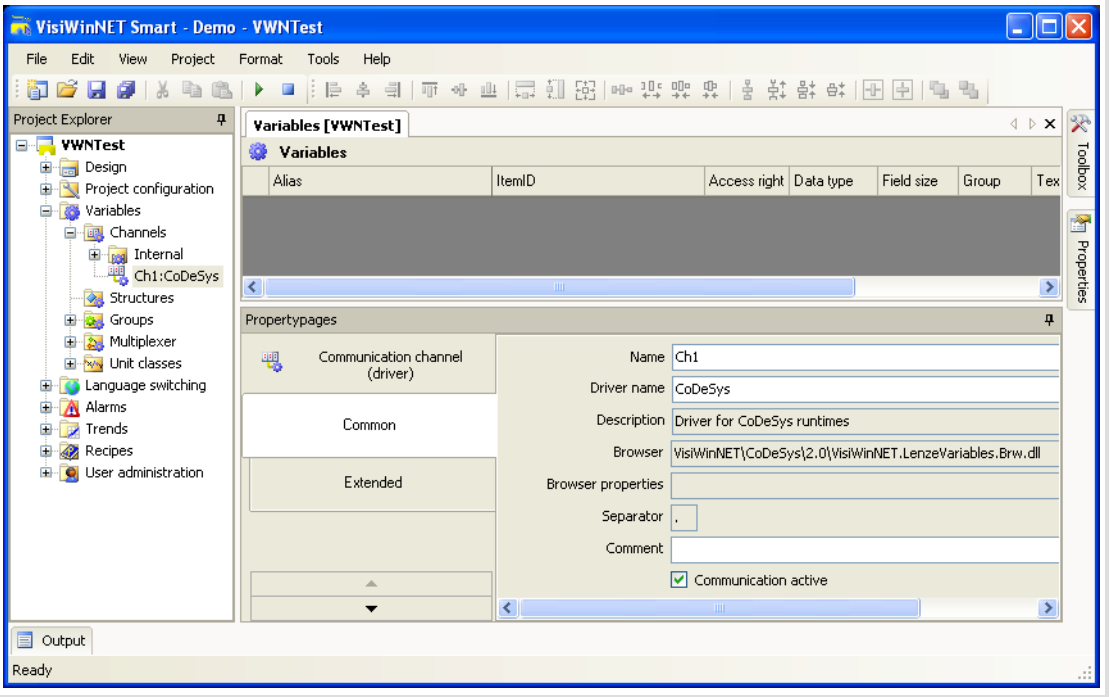
Configuring a channel

Proceed as follows:

1. Go into Project Explorer and create a new channel: Highlight the **Variables / Channels** folder, right-click it and then select **New**.
2. In the line **OPC-Server/Driver**, click **[...]** and select **CoDeSys**.



3. Close the dialog with **OK**.
   The channel is created and displayed in the project tree.
4. Double-click the new channel in the object tree and configure it on the Properties page.



Line: A line is a communication relationship between the EL1xx and an L-force Logic (CoDeSys) application. A maximum of ten lines can be set up, i.e. an EL1xx PLC can exchange data with a maximum of 10 L-force Logic (CoDeSys) applications.

Line active: The line is activated by specifying TRUE in the „Set value“ field and is deactivated by specifying FALSE.

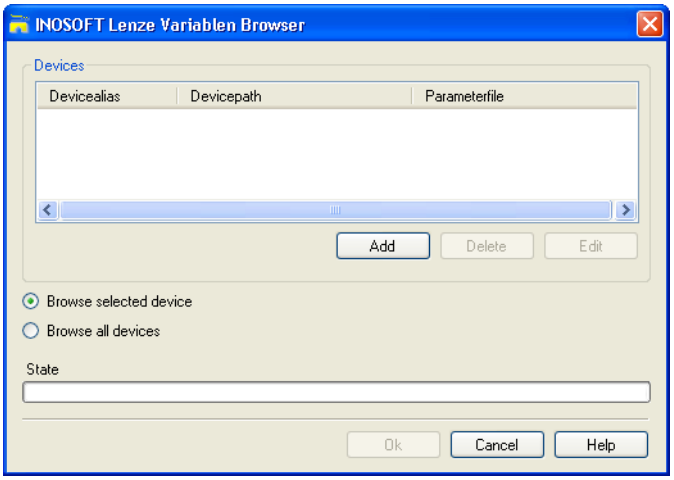
PLC Port: TCP/IP port used by the PLC.

IP: IP address of the PLC runtime system. The IP address of the runtime system corresponds to the IP address of the EL1xx PLC.

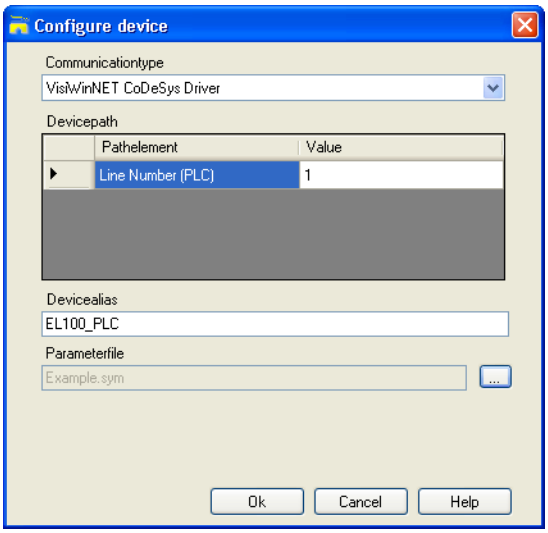
PLC is local: If TRUE has been specified, the VisiWinNET®Smart application uses the local PLC runtime system. The settings under ”IP” are ignored.

Browsing variables

5. Highlight the **CoDeSys channel** in Project Explorer, right-click it and select **Browse variables**.



6. Click **Add** and fill in the "Configure device" dialog.



Communication type: Communication type "VisiWinNET CoDeSys Driver"

Device path: Specification of the line, i.e. the CoDeSys application with which the browsed variables are to be exchanged.

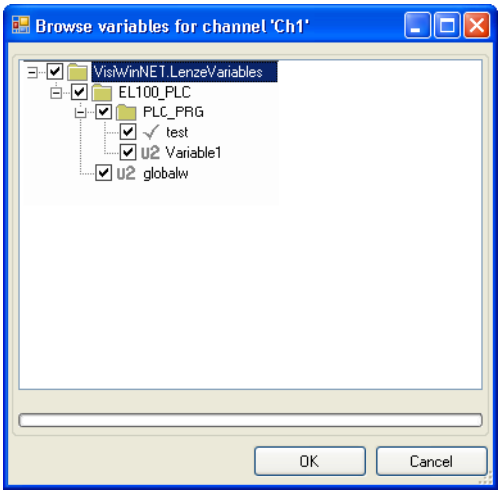
Device alias: Enter any name you like.

Parameter file: Name of the sym file that can be generated by the L-force Logic (CoDeSys) and is needed for visualisation of the PLC program on the EL 1xx PLC (📖 250). The sym file for the PLC program is located in the L-force Logic project folder (CoDeSys).

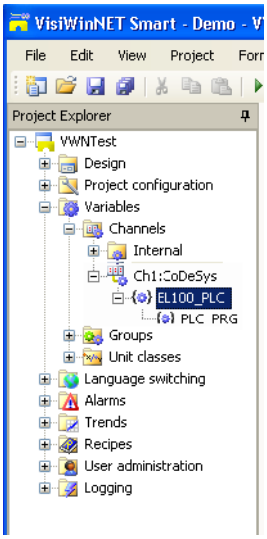
7. Close the dialog with **OK**.

Reading in variables

8. Select the device from which the variables are to be read.



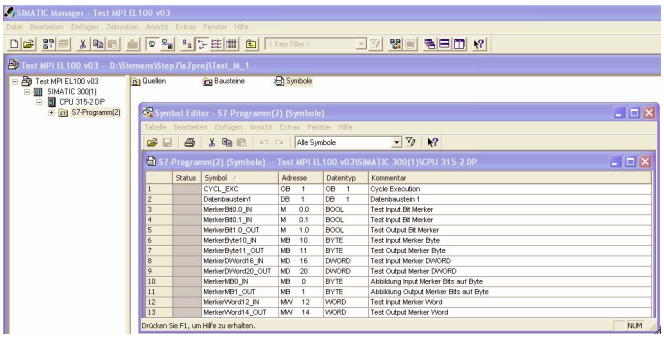
9. Close the dialog with **OK**.
10. Select the variables to be imported from the following dialog.
11. Close the dialog with **OK**.
    The variables selected are now available in VisiWinNET®Smart.



6 Appendix

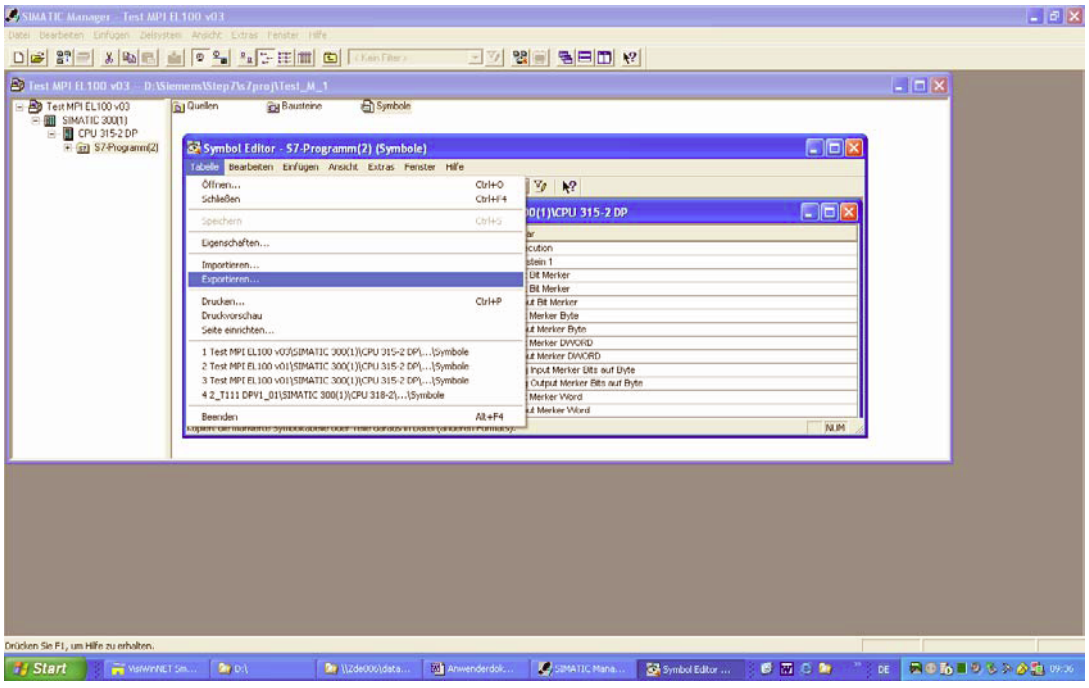
6.1 Generating an sdf file from a symbol table

1. Go to the ”Simatic Manager” and open the symbol table with a doubleclick on the ”Symbol” icon.



VWNS016

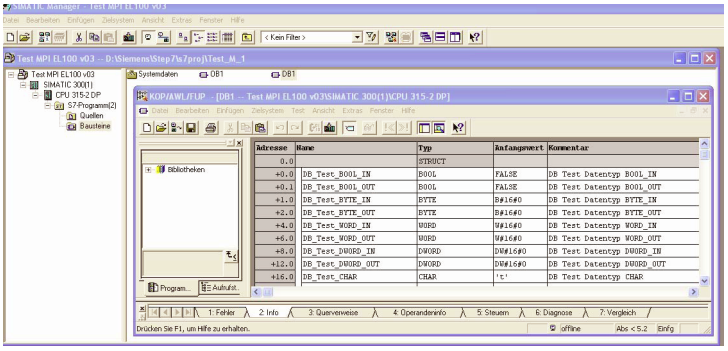
2. Select **Table** → **Export** to open the ”New” dialog, enter the file name in the ”Object name” field and confirm with **OK**.



VWNS017

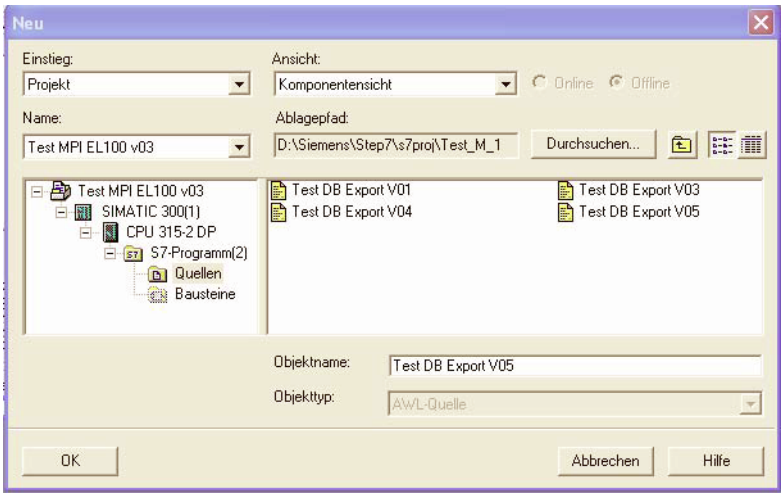
6.2 Create instruction list file from a data block

1. Go to the ”Simatic Manager” and double-click the DB icon to open the data block.



VWNS012

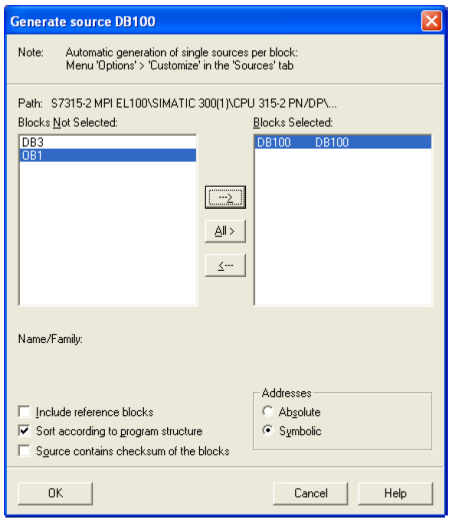
2. Select **File** → **Generate source** to open the ”New” dialog. Enter the file name in the ”Object name” field and confirm with **OK**.



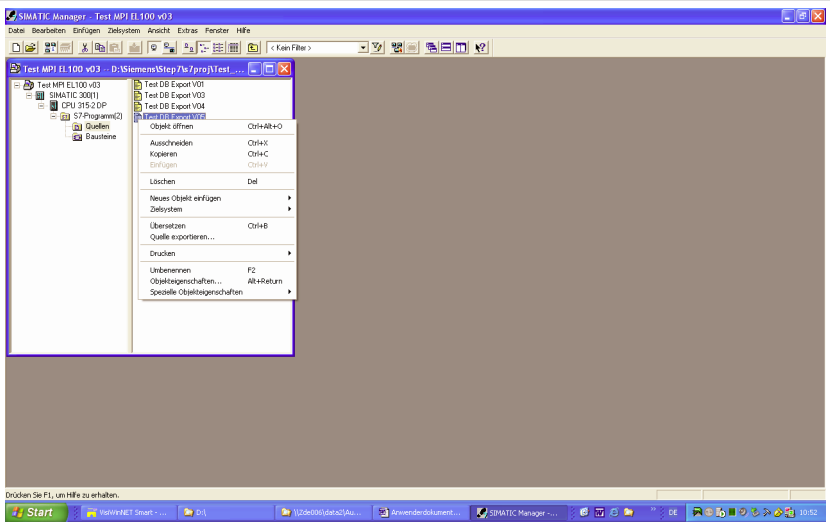
VWNS013

3. Select blocks that are to be contained in the source and highlight the entry ”Symbolic” in the ”Operands” option field.

   These blocks can then be used in VisiWinNET.

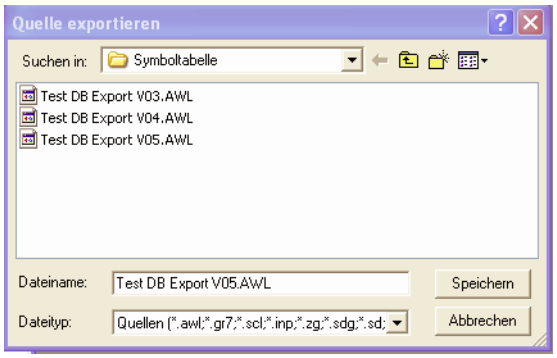


4. Select the generated source in the project tree, click the right mouse button and activate **Export source**.



VWNS014

5. Enter a name for the source to be exported under ”File name” and select ”.awl” as ”Data type”. Complete your selection with **Store**.



VWNS015

The stored instruction list file can now be read in in VisiWinNET.

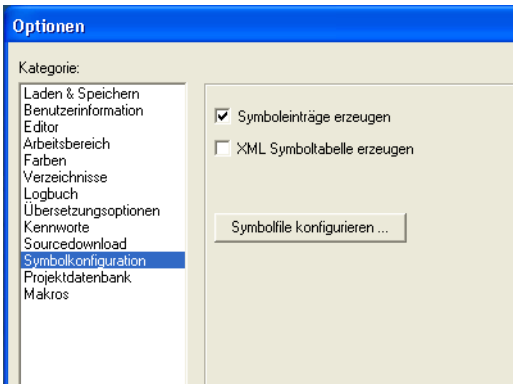
Symbol activation takes place in the development environment of the SimaticManager.

6.3 Creating the sym file in the PLC Designer

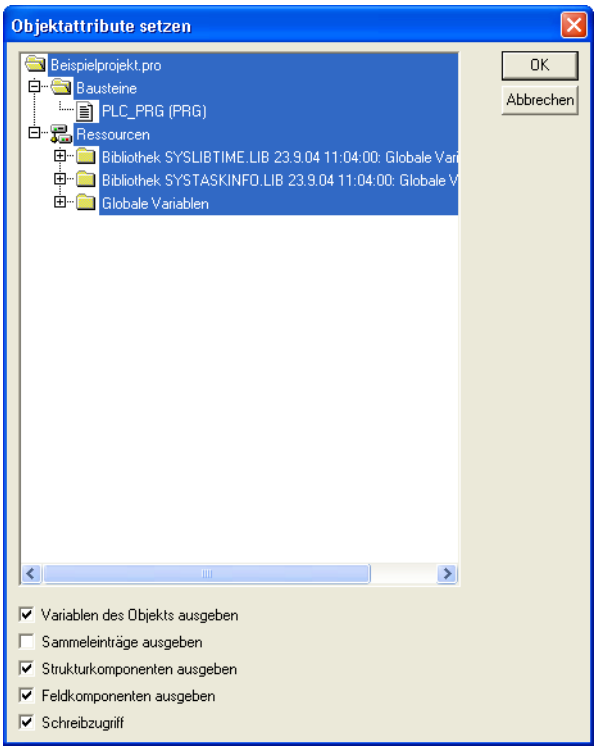
A sym file is required if the visualisation, e.g. VisiWinNet® Smart is to access the PLC program. A sym file contains all variables and declarations of the PLC program.

How to proceed:

1. Open the menu **Project** in the PLC Designer and select the menu item **Options**.
   The ”Options” dialogue will be opened.



2. Select the ”Symbol configuration” category from the ”Options” dialogue.
3. Select the **Dump symbol entries** control field and click the **Configure symbol file** button.
   The ”Set object attributes” dialogue will be opened.



4. Select the *Export variables of object* control field in the ”Set object attributes” dialogue. Do not change the other control field selections.
5. Go to the object tree and select all objects which contain variables for the visualisation and are enabled via the variables of the object.

   Only export those variables required for visualisation. The smaller the sym file the shorter the baud rate to the target system.
6. Close the ”Set object attributes” dialogue and the ”Options” dialogue with **OK**.
7. Go to the menu bar, open the menu **Project** and select the menu item **Clean all**.
8. Open the menu **Project** once again and select the menu item **Rebuild all**.

   The project is checked, compiled and saved under the file name ”ExampleProject.sym” at the selected memory location.

6.4 Scripts

6.4.1 Basics

VisiWinNET® Smart supports the creation of an individual program logic by the programming of user-defined functions using the programming language VB.NET. The functions required, which are generally also referred to as scripts, can be stored in different code modules via an integrated code editor.

In contrast to a standard script, however, the functions created with VisiWinNET® Smart are not interpreted during the runtime of the program, but translated into an executable .NET code already during the project creation. This produces a higher execution speed compared to that of classical scripts and furthermore has the advantage that the entire functional range of the .NET framework within the code modules can be used.

A code module can either be stored globally or be assigned to a form.

Globally stored code module

In global code modules functions or global variables are stored, which are required in different positions within the program.

A global code module can be added via the context menu of the project explorer node **Design**:

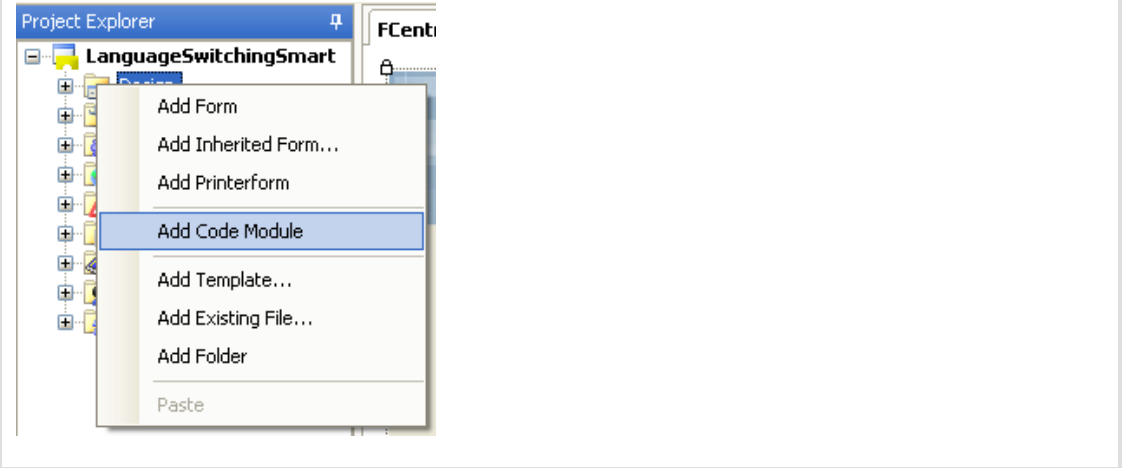


Fig. 6-1 Add code module

Code module assigned to a form

Code modules which are assigned to a form contain functions that are required within a specific form. The events of the form and of the control elements and components located on it can be trapped within the code module.

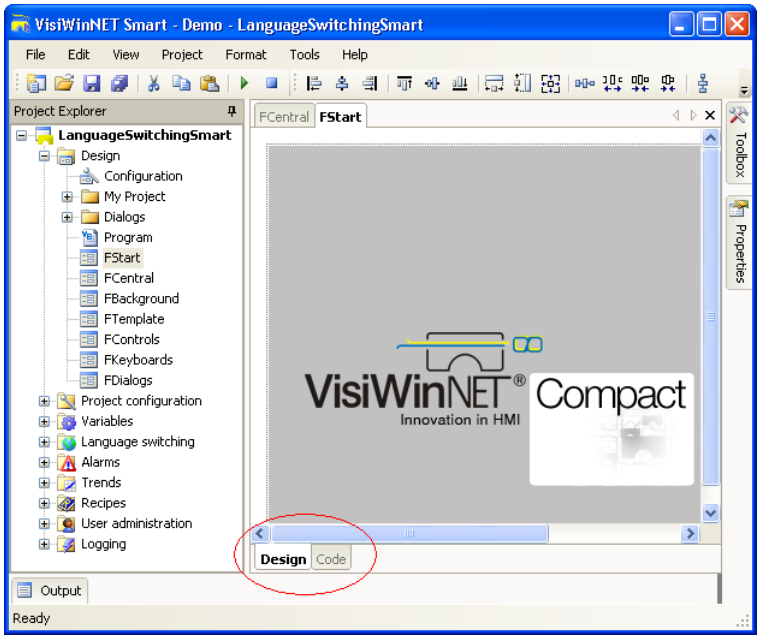


Fig. 6-2 Switching over between the form view (design) and the code view (code)

Structure of scripts

Events are used to call a function. In the code module an event is represented by a frame function into which the executable code is written. The insertion of a frame function is supported by the two selection lists in the upper position of the code module.

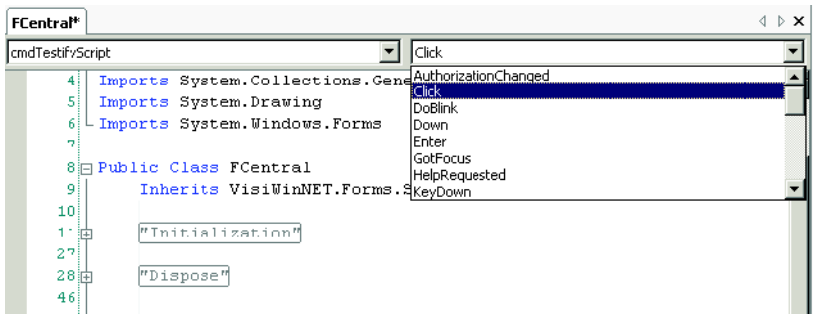


Fig. 6-3 Example code module

The left selection list contains the names of all objects which can trigger events, i.e. the names of the form and of the control elements and components on it. After selecting a name the selection list on the right itemises all events that are supported by the object selected on the left.

Example: The CommandButton control element selected on the left with the name ”cmdTestifyScript” among other things also provides a ”Click” event which is triggered if the user clicks on the control element with the left mouse button.

Inserting a frame function

By selecting an event name on the right a frame function is inserted.

Example of an event frame function:

```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles VarIn1.Change

End Sub
```

The selected object ”VarIn1” provides the ”Change” event which signalises the change of the variable value. For inserting the function the object ”VarIn1” was selected in the selection list on the left, afterwards in the ”Change” selection list on the right.

The function button of the frame function inserted contains the following:

- A name for the function (”VarIn1_Change”). The name has to be unambiguous within a code module.
- The parameters ”sender” and ”e”; ”sender” usually being a reference to the object within which the event has been triggered. ”e” usually contains further information with regard to the event.
- The ”Handles” condition describes the objects and events by which the function is triggered. In this case this is only a Change event of VarIn1. However, you can switch further events with identical parameters to the function by adding further event identifiers (<Object name>.<Event name>), separated by commas, to the ”Handles” condition. In this case it becomes clear why the ”Sender” parameter is included in the function. Via ”Sender” the object which has triggered this event can then be determined.

The function end is represented by ”End Sub”. Between the function head and the function end the executable VB code is permissible:

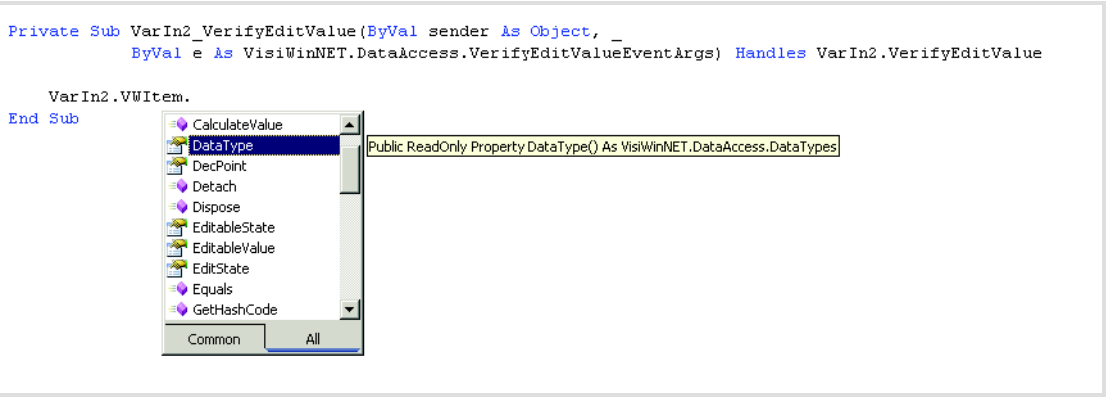
```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles VarIn1.Change

    If e.Value > 50 Then
      VarIn1.DataBackColor = System.Drawing.Color.Red
    Else
      VarIn1.DataBackColor = System.Drawing.SystemColors.Window
    End If

End Sub
```

Example: Within the e-argument further information with regard to the event is provided. Here the background colour in the data area of ”VarIn1” is switched over on the basis of the value property from the variable core.

Help for the creation of scripts

VisiWinNET® Smart contains programming interfaces for the surface elements and the underlying system.

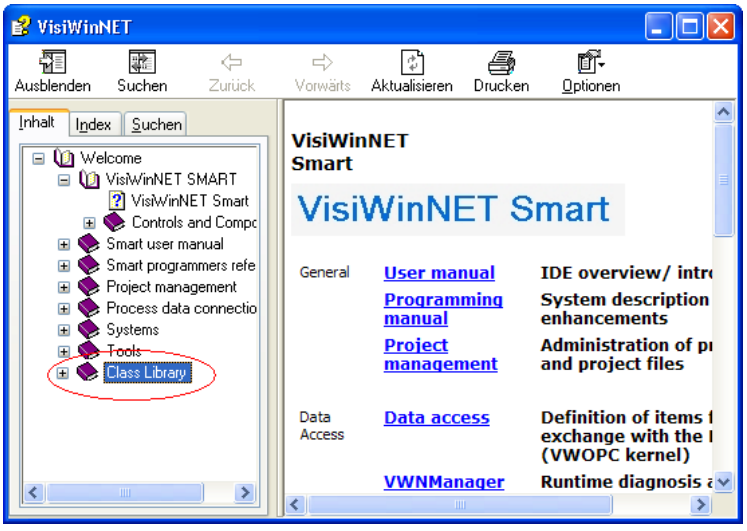
You can obtain information in brief form about the functions (properties/methods/events) provided by an object by means of the Intellisense window which is implemented in the source code editor and which opens automatically when object names or name areas are entered (the use of upper/lower case is important!).



Via the specification of the object name ”VarIn2” with a concluding ”.” all properties are listed. Properties are objects and in turn include properties which are selected via the cursor keys and inserted by the Enter key.

By specifying the word ”VisiWinNET” you are lead into the name areas containing all visualisation functions ordered by subjects.

Additionally a programming reference for research is provided in the online help, which describes the interfaces of the product from the programmer’s view.



Tip!

A recommendable introduction into programming with VB can be found under ”http://openbook.galileocomputing.de/vb_net/”. The applicable integrated development environment for the introductory course mentioned is offered free of charge under the name ”Visual Basic Express Edition” by Microsoft.

In the following chapter ”Example functions” you can furthermore find a choice of examples which according to experience are often used in projects.

The .NET framework provides a variety of functions supporting the common programming tasks under Windows. By using a search engine it is often possible to obtain a small piece of functioning source code, which works with the framework. However, in most cases you additionally have to integrate a reference to the corresponding assemblies in the project. This is carried out in the project explorer via the node **Design** → **References**.

During research the ”MSDN online” (http://msdn.microsoft.com/de-de/default.aspx) helps, containing both information on the programming language and on the .NET framework.

6.4.2 Troubleshooting within functions

Generally two types of errors are distinguished:

- Syntactic errors
- Runtime errors

Syntactic errors

Syntactic errors or missing variable declarations are already reported during the creation of the project. Detecting and eliminating these errors is relatively uncomplicated for the developer, since they are already reported during the development stage. Programming errors in the project are shown during the creation (via the menu item **Project → Creation**" or the shortcut **Ctrl+Shift+B**)in the output window. If the project contains errors that are detected by the compiler, it cannot be started.

Runtime errors

Runtime errors such as "Division By Zero" or missing object references are only detected when the corresponding function is executed. The risk with this is that after commissioning the code is supplied with data from the process which have not been checked and result in an exception. Here it helps to use the Try-Catch condition with a defined response for the case of error.

```
Private Sub ItemServer1_Change(ByVal sender As System.Object, _
    ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) Handles ItemServer1.Change
  Try
    Dim Divident As Double = _
      VisiWinNET.Services.AppService.VWGet("Ch1.myDivident")

    Dim Divisor As Double = e.Value

    Dim Quotient As Double = Divident / Divisor

    VisiWinNET.Services.AppService.VWSet("Ch1.myQuotient", Quotient)

  Catch ex As Exception 'ooh, an error
    If MsgBox("ItemServer1_Change results in: " _
          & vbCrLf & ex.Message _
          & vbCrLf & "Continue?", _
           MsgBoxStyle.YesNo) = MsgBoxResult.No Then

      'decision: End application
      VisiWinNET.Forms.ProjectForms.StartForm.ShutdownProject()

    End If
  End Try
End Sub
```

Example: If a runtime error during the calculation "dividend/divisor= quotient" occurs, the program run skips to the Catch condition. By means of a message box it is now decided whether the application is to be closed.

6.4.3 Example functions

Reading and writing of variable values

A typical scenario is the reading of one or several variables if a trigger is activated. The trigger often is a variable itself. The reading/writing is executed via the function VWGet/VWSet.

```
'Sample VWGet/VWSet: Reading a byte field from Demo OPC Server
'increment elements and write back
Private Sub ItemServerArrayRead_Change(ByVal sender As System.Object, _
        ByVal e As VisiWinNET.DataAccess.ChangeEventArgs)
        Handles ItemServerArrayRead.Change
    If e.Value = 1 Then
      Dim xObj As Object = _
        VisiWinNET.Services.AppService.VWGet("Ch1.Simulation.Array.Byte")
      Dim x() As Byte = xObj
      For i As Integer = 0 To x.Length - 1
        Dim bt As Byte = x(i)
        bt = bt + 1
        x.SetValue(bt, i)
      Next
      VisiWinNET.Services.AppService.VWSet("Ch1.Simulation.Array.Byte", x)
    End If
  End Sub
```

During the course of a Change event, this function reads out the field "Simulation.Array.Byte" from a demo OPC server from the variable core of a VarServer component, increments the elements and writes the data field back. Further examples with regard to the functions "VWGet"/"VWSet" can be found in the online help.

Evaluating bits in integer variable values

VisiWinNET® Smart contains functions which support the VB programmer during handling bits from integer variable values.

```
'Toggles Bit 2 in process variable "MyBitModulatedValue"
If VisiWinNET.Services.AppService.VWGetBit("MyBitModulatedValue", 2) = True Then
    VisiWinNET.Services.AppService.VWSetBit("MyBitModulatedValue", 2, False)
Else
    VisiWinNET.Services.AppService.VWSetBit("MyBitModulatedValue", 2, True)
End If
```

This function accesses bit 2 in the "MyBitModulatedValue" process variable and changes the value.

Output information for checking

In order to facilitate diagnostics in spite of the missing debugger, VisiWinNET® Smart diverts standard outputs to the output window. Standard outputs are shown line by line via the command Console.WriteLine.

```
Dim i As Integer = 21
Console.WriteLine(”Hello World, the answer is: ” & i * 2)
```

The output is displayed in the output window of VisiWinNET® Smart.

Under Windows CE, the following code must be used:

```
Private i As Integer = 0

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
   #if DEBUG
    Console.WriteLine(”TimerTick: ” & i)
      i = i + 1
   #End If
End Sub
```

The Compiler switch can be set under Design\Configuration\Build.

Reading texts from the language switching

Via the types in the name area ”LanguageSwitching” the information from the language switching can be obtained. This also includes the reading of individual texts.

```
Dim s As String =_
       VisiWinNET.LanguageSwitching.Localization.GetText(”@FCentral.MyText”)
```

In the language switching editor, a text group ”FCentral” was created, and within it a text called ”MyText”. After the function call the text is provided in the language currently set in the project in variable ”s”.

A smaller amount of code for a great number of functions (global modules)

The introductory example for switching the background colour of a ”VarIn” control element as a function of the variable value in the chapter ”Scripts” can be extended to several control elements on different forms by using a global code module.

For this, insert a new code module (”Module1”) in the project via the context menu of the ”Design” project explorer node. Insert the following function in this code module:

```
Public Sub SetVarinDataBackColor(ByVal VarIn As VisiWinNET.Forms.VarIn, _
            ByVal e As VisiWinNET.DataAccess.ChangeEventArgs)
    If e.Value > 50 Then
        VarIn.DataBackColor = System.Drawing.Color.Red
    Else
        VarIn.DataBackColor = System.Drawing.SystemColors.Window
    End If
End Sub
```

The triggering ”VarIn” object and the ”ChangeEventArgs” object transferred in the Change event are expected as function parameters.

The call of the function from a Change event thus is reduced to one single line and then looks like this:

```
Private Sub VarIn1_Change(ByVal sender As System.Object, _
            ByVal e As VisiWinNET.DataAccess.ChangeEventArgs) _
            Handles VarIn1.Change
    Module1.SetVarinDataBackColor (sender, e)
End Sub
```

Advantages of the use of the global function:

- A smaller amount of code during the call
- Better parameterisation possible (e.g. if the limit value ”50” is to be altered centrally)

6.4.4 Restrictions in the programming interface of VisiWinNET Smart

The programming interface of VisiWinNET® Smart has the following restrictions:

- VisiWinNET®Smart does not contain an individual debugger. Errors are either shown during the creation or result in an exception during runtime. If in especially persistent cases debugging appears to be inevitable, the CLR debugger (component of the Microsoft .NET framework SDK 2.0) can be used. For projects below Windows CE, even this possibility is no longer provided, since the CLR debugger does not supply remote debugging support.
- More complex tasks such as the development of individual control element or class libraries due to the missing comfort functions (control element designer, organisation of several projects within one solution, debugging during the development time...) are reserved for the professional programmer. For Visual Studio the functions mentioned are provided.

6.5 Translation manager

The development of a multilingual visualisation application includes the translation of all texts used in the application. For this often an external translator is required.

As a virtual standard for exchanging the texts between the project manager and the translator, VisiWinNET® Smart is provided with the translation manager. Here Excel lists can be created, containing the texts from the VisiWinNET® Smart project, which are to be translated. After the translation process the contents of the files exported can be inserted in the project again.

Translators have to be equipped with Excel by Microsoft Office from the ”Office 2000” version onwards.

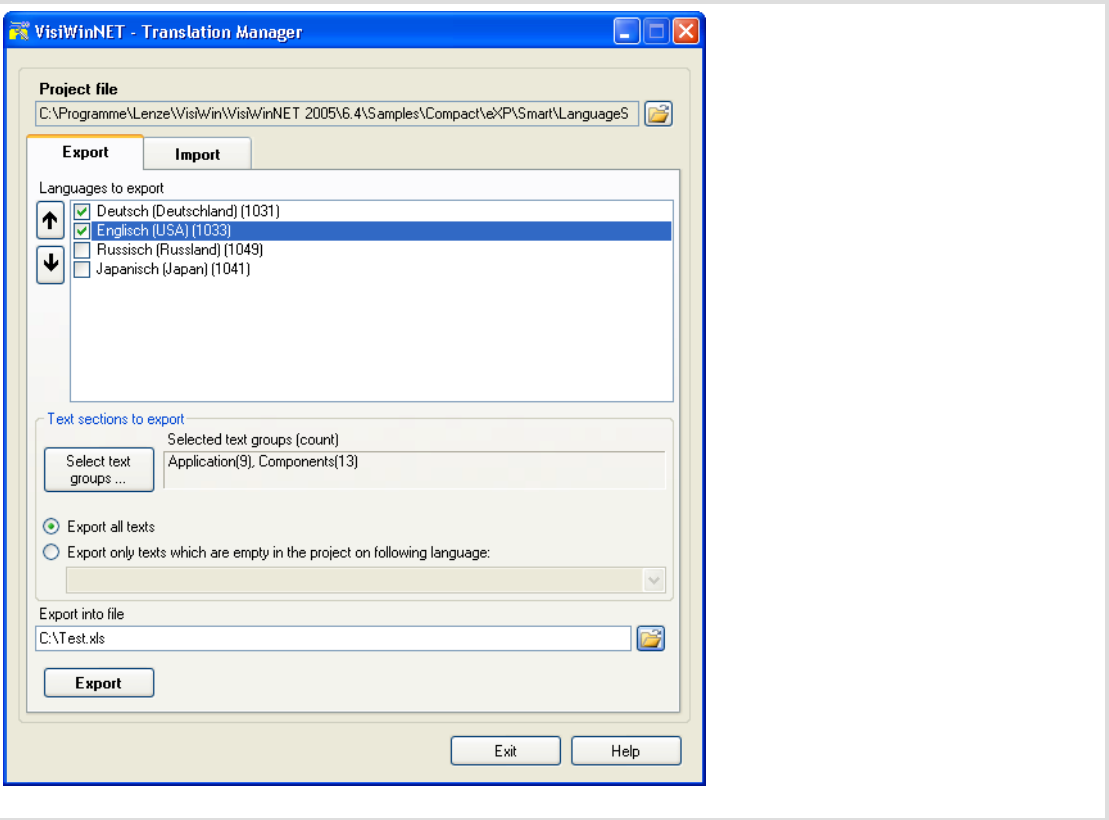
The translation manager is started via the Windows Start Menu. During the installation of VisiWinNET®Smart a corresponding entry is inserted in the **VisiWin** → **VisiWinNET 2005** program group.

The project manager’s task is to export the texts to be translated as an Excel file with the aid of the translation manager, to protect them, and to send them to the translator. After the translation has been completed, the project manager has to re-import the information received.

6.5.1 Exporting texts

During the export process an Excel file is created, which contains the texts from the project that are to be translated and that can be switched to other languages. This file can be sent to the translator.

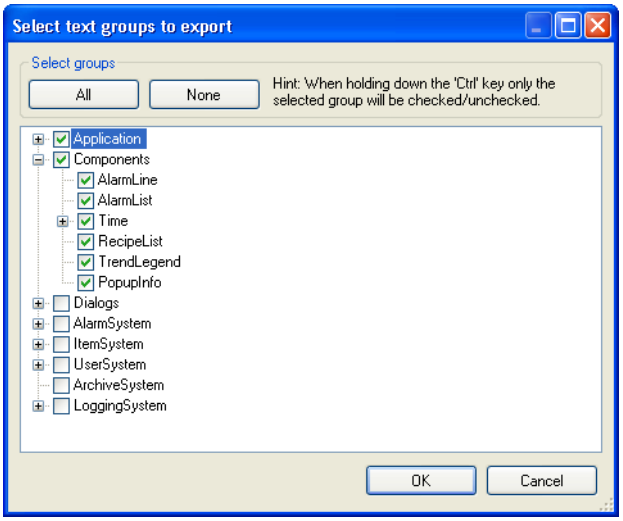
The settings and functions for the export process are located within the translation manager on the file card with the same name.



Project file: Serves to select a VisiWinNET® Smart project database (*.vwn) from which the texts are to be exported. Via the button on the right a file selection dialog is opened.

Languages to be exported: Lists the languages of the project. A maximum of three languages can be selected for the export process. The first language serves as source language of the translation. The second language can serve as an auxiliary language. The last language is the language which is to be completed by the translator. Via the arrow buttons the sequence of the languages to be exported is set.

Text areas to be exported: The export can be filtered according to individual text groups. The selection dialog for the text groups is opened via the ”Select text groups” button.



The selection dialog contains the following control elements:

- By clicking on the selection box in the group view the group and all the subgroups are selected/deselected. If the Ctrl key remains pressed during this selection, the selection status of the subgroups does not change.
- The ”All” button highlights all text groups shown.
- The ”None” button unmarks all groups.
- OK confirms the selection.
- ”Cancel” closes the dialog without accepting the selection.

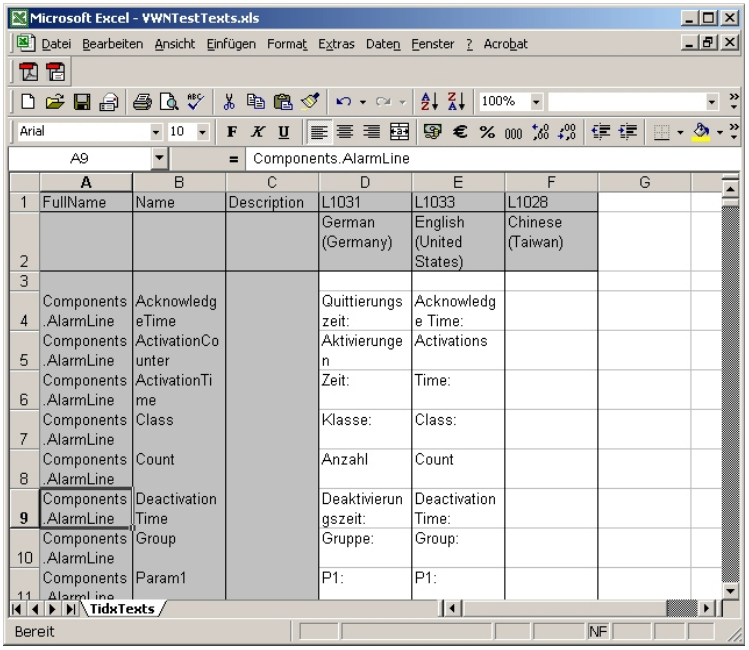
Import all texts/ only import texts which are empty in the project for the following language: Option for the repeated translation. If a language has already been translated once, the option ”Only import texts which are empty in the project for the following language” serves to only export new texts (i.e. texts that were not there during the last translation). The following list offers the selection of a language the content of which is to be checked with regard to empty texts.

Export to file: Specifies the name of the file to be newly generated via a file selection dialog (button).

Export: With this button the export process is started.

Close: Closes the program.

6.5.2 Protecting the export file



	A	B	C	D	E	F	G
1	FullName	Name	Description	L1031	L1033	L1028	
2				German (Germany)	English (United States)	Chinese (Taiwan)	
3							
4	Components.AlarmLine	AcknowledgeTime		Quittierungszeit:	Acknowledge Time:		
5	Components.AlarmLine	ActivationCounter		Aktivierungen	Activations		
6	Components.AlarmLine	ActivationTime		Zeit:	Time:		
7	Components.AlarmLine	Class		Klasse:	Class:		
8	Components.AlarmLine	Count		Anzahl	Count		
9	Components.AlarmLine	DeactivationTime		Deaktivierungszeit:	Deactivation Time:		
10	Components.AlarmLine	Group		Gruppe:	Group:		
11	Components.AlarmLine	Param1		P1:	P1:		

The Excel file generated by the translation manager contains the worksheet ”TidxTexts” with the exported texts. The first two lines present the table head. Then the texts follow line by line, the first two columns ”FullName” and ”Name”, respectively, specifying the complete path of the corresponding text, which is important for the re-import process.

The Excel file contains technicalities which must not be changed by the translator.

- The table head (the first two lines) must not be changed or deleted by the translator.
- Also with regard to the first two columns the translator always has to observe: Do not change.
- The ”Description” column contains the ”description” for the text. If it is filled out reasonably by the project manager, it can be a valuable help for the translator. The contents, however, are ignored during the re-import process.
- If the option ”Only import texts that are identical in the project and the import file” is to be used for the re-import process, the translator must not change the field contents in the language used for purposes of comparison.

Note!

It quickly happens that one of the above-mentioned fields is accidentally changed or deleted by the translator. Take precautions for this:

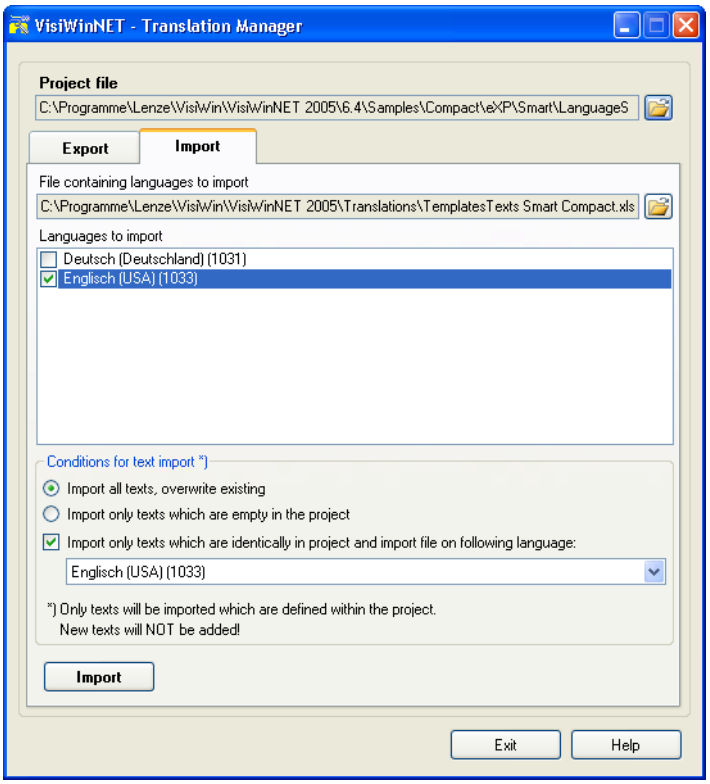
Excel offers the possibility of blocking out cells of a worksheet. For this the corresponding cells first are to be highlighted (by lines or columns). Via the menu ”Format”—>”Cells” a corresponding option can be activated on the ”Protection” tab. Afterwards a password has to be assigned to the worksheet. Like this, only the fields that have not been blocked out can be changed on the worksheet. Further information on this can be found in the Office online help.

6.5.3 Importing texts

The import process serves to adopt the texts, which the translator has returned, from the Excel list to the project database. The settings and functions for the import process are located in the translation manager on the file card with the same name.

Note!

The import is limited to the texts in the project database. If the Excel table is extended by new lines, the texts which have been generated like this are not inserted in the project database. The translation manager thus is no alternative to the language switching editor. If you want to predefine the texts within a table outside the editor and then import them, use the possibility of insertion via the Windows clipboard instead. The VisiWinNET® Smart editors provide the possibility of working via the "Copy" and "Paste" functions with an Excel-compatible clipboard format.


VisiWinNET - Translation Manager
Project file
C:\Programme\Lenze\VisiWin\VisiWinNET 2005\6.4\Samples\Compact\eXP\Smart\LanguageS
Export
Import
File containing languages to import
C:\Programme\Lenze\VisiWin\VisiWinNET 2005\Translations\TemplatesTexts Smart Compact.xls
Languages to import
Deutsch (Deutschland) (1031)
Englisch (USA) (1033)
Conditions for text import *)
Import all texts, overwrite existing
Import only texts which are empty in the project
Import only texts which are identically in project and import file on following language:
Englisch (USA) (1033)
*) Only texts will be imported which are defined within the project.
New texts will NOT be added!
Import
Exit
Help

The dialog features the following control elements:

Project file: Serves to select the VisiWinNET® Smart project database (*.vwn) into which the translated texts are to be imported. Via the button on the right a file selection dialog is opened.

File with language to be imported: Serves to select the Excel file (*.xls) which contains the texts returned by the translator. Via the button on the right a file selection dialog is opened.

Note!

The VisiWinNET® Smart system texts are available in the standard languages German and English. You will also find system texts in French, Italian and Spanish in the folder "...\VisiWinNET 2005\Translations" which you can incorporate in your application with the help of the Translation manager.

Languages to be imported: Selects the languages the texts of which are to be re-imported in the project.

Import all texts, existing texts are overwritten: Specifies that all texts already existing in the project are to be overwritten in the languages selected. This option is reasonable if a translation has already been imported in the project and the translator has now carried out further adaptations to the texts already imported.

Only import texts that are empty in the project: Specifies that only the empty texts in the project are to be overwritten in the language selected. This option is reasonable if the texts in the project have already been checked.

Only import texts that are identical in the project and the import file for the following language: The selection list below the control element offers a language the texts of which are compared to each other. Only if the text of this language is the same in the external file and in the project, the corresponding text is re-imported. Usually the source language of the translation is specified here. If the texts of the source language for instance have been edited after the export process, setting this option can avoid that translations which are now out-of-date are imported in the project.

Import: With this button the re-import process is started.

Close: Closes the dialog.

7 Index

A

Alarm system, 205

Apply a language, 200

Archive system, 215

B

Browse variables, 232 , 244

Browsing variables, 175 , 239

C

CAN
- Browsing variables, 232
- Configure channel, 231 , 235
- Configure connection
  Via codes, 231
  via PDOs, 234
- Creating PDO variables, 236
- PDO-specific settings, 236
- Reading in variables, 233

Client area, 144

CoDeSys
- Configuring a channel, 242
- Configuring a connection, 242
- Reading in variables, 245

Configuration, 226

Configuring a channel, 231 , 235 , 238 , 242

Context menu, 150

Create PDO variables, 236

Create sym file, 250

Creating a channel, 173

Creating a communication channel, 173

Creating a visualisation application, 176

Creating an alarm, 209

Creating an alarm group, 207

Creating forms, 176

D

Defining a process variable, 206

Displaying values, 176

Dongle, 140

Dynamic texts, 202

E

EL 100, 141

Emulator
- Windows CE, 160
- with active communication, 161
- with variable simulation, 161

F

Filtering alarm displays, 214

Form designer, 152

H

Help, 145

HMI EL 100, 141

I

Installation, 142

Instruction list file, 247

L

L-force Logic
- Configuring a channel, 242
- Configuring a connection, 242
- Reading in variables, 245

Language settings, 228

Language switching, 194

Licence terms, 140

M

Menu bar, 144 , 157

MPI
- Browse variables, 239
- Configure a channel, 238
- Configure connection, 238
- Read variables, 241

O

Operation, 143

Output window, 144 , 154

Outputttting project data, 223

P

PDO-specific settings, 236

PLC Designer
- Browsing variables, 244
- Create sym file, 250

PLC program, Create sym file, 250

Printing, 223

Printing information, 223

Program
- exit, 149
- install, 142
- interface, 143
- start, 143

Project
- Configure, 226
- converting, 149
- Create, 168
- create, 147
- Create in several languages, 194
- Example, 168
- Open, 146
- Save, 148
- Settings, 226
- Transfer, 229

Project example, 168

Project Explorer, 150

Project Explorer , 144

Properties window, 144 , 153

Property page, 156

Property pages, 144

R

Read in variables, 233

Reading in variables, 245

Reading variables, 241

Recipes, 219

Record process data, 215

Regional and language settings, 200

S

Scope of supply, 140

sdf file, 246

Specifying the type of alarm acknowledgement, 211

System requirements, 141

System texts, 266

T

Table editor, 155

Testing the alarm system, 213

Texts
- dynamic, 202
- user, 196

Toolbar
- Format, 159
- Standard, 159

Toolbars, 144

Toolbox, 144 , 153

Translation manager, 261

Trend, 215

U

User administration, 184

User texts, 196

Using control elements, 176

V

Visualisation recipes, 219

Visualising alarms, 212

© 03/2012

Lenze Automation GmbH
Hans-Lenze-Str. 1
D-31855 Aerzen
Germany

+49 (0)51 54 / 82-0

+49 (0)51 54 / 82 - 28 00

Lenze@Lenze.de

www.Lenze.com

Service

Lenze Service GmbH
Breslauer Straße 3
D-32699 Extertal
Germany

00 80 00 / 24 4 68 77 (24 h helpline)

+49 (0)51 54 / 82-11 12

Service@Lenze.de

SHP-VWNS ■ 13404124 ■ DE/EN ■ 6.1 ■ TD29

10 9 8 7 6 5 4 3 2 1